Preface: Digital Signal Processing and Digital Filter Design

Digital signal processing (DSP) has existed as long as quantitative calculations have been systematically applied to data in Science, Social Science, and Technology. The set of activities started out as a collection of ideas and techniques in very different applications. Around 1965, when the fast Fourier transform (FFT) was rediscovered, DSP was extracted from its applications and became a single academic and professional discipline to be developed as far as possible.

One of the earliest books on DSP was by Gold and Rader [link], written in 1968, although there had been earlier books on sampled data control and time series analysis, and chapters in books on computer applications. In the late 60's and early 70's there was an explosion of activity in both the theory and application of DSP. As the area was beginning to mature, two very important books on DSP were published in 1975, one by Oppenheim and Schafer [link] and the other by Rabiner and Gold [link]. These three books dominated the early courses in universities and self study in industry.

The early applications of DSP were in the defense, oil, and medical industries. They were the ones who needed and could afford the expensive but higher quality processing that digital techniques offered over analog signal processing. However, as the theory developed more efficient algorithms, as computers became more powerful and cheaper, and finally, as DSP chips became commodity items (e.g. the Texas Instruments TMS-320 series) DSP moved into a variety of commercial applications and the current digitization of communications began. The applications are now everywhere. They are tele-communications, seismic signal processing, radar and sonar signal processing, speech and music signal processing, image and picture processing, entertainment signal processing, financial data signal processing, medical signal processing, nondestructive testing, factory floor monitoring, simulation, visualization, virtual reality, robotics, and control. DSP chips are found in virtually all cell phones, digital cameras, high-end stereo systems, MP3 players, DVD players, cars, toys, the “Segway", and many other digital systems.

In a modern curriculum, DSP has moved from a specialized graduate course down to a general undergraduate course, and, in some cases, to the

introductory freshman or sophomore EE course [link]. An exciting project is experimenting with teaching DSP in high schools and in colleges to non-technical majors [link].

Our reason for writing this book and adding to the already long list of DSP books is to cover the new results in digital filter design that have become available in the last 10 to 20 years and to make these results available on line in Connexions as well as print. Digital filters are important parts of a large number of systems and processes. In many cases, the use of modern optimal design methods allows the use of a less expensive DSP chip for a particular application or obtaining higher performance with existing hardware. The book should be useful in an introductory course if the students have had a course on discrete-time systems. It can be used in a second DSP course on filter design or used for self-study or reference in industry.

We first cover the optimal design of Finite Impulse Response (FIR) filters using a least squared error, a maximally flat, and a Chebyshev criterion. A feature of the book is covering finite impulse response (FIR) filter design before infinite impulse response (IIR) filter design. This reflects modern practice and new filter design algorithms. The FIR filter design chapter contains new methods on constrained optimization, mixed optimization criteria, and modifications to the basic Parks-McClellan algorithm that are very useful. Design programs are given in MatLab and FORTRAN.

A brief chapter on structures and implementation presents block processing for both FIR and IIR filters, distributed arithmetic structures for multiplierless implementation, and multirate systems for filter banks and wavelets. This is presented as a generalization to sampling and to periodically time-varying systems. The bifrequency map gives a clearer explanation of aliasing and how to control it.

The basic notes that were developed into this book have evolved over 35 years of teaching and conducting research in DSP at Rice, Erlangen, and MIT. They contain the results of research on filters and algorithms done at those universities and other universities and industries around the world. The book tries to give not only the different methods and approaches, but

also reasons and intuition for choosing one method over another. It should be interesting to both the university student and the industrial practitioner.

We want to acknowledge with gratitude the long time support of Texas Instruments, Inc., the National Science Foundation, National Instruments, Inc. and the MathWorks, Inc. as well as the support of the Maxfield and Oshman families. We also want to thank our long-time colleagues Tom Parks, Hans Schuessler, Jim McClellan, Al Oppenheim, Sanjit Mitra, Ivan Selesnick, Doug Jones, Don Johnson, Leland Jackson, Rich Baraniuk, and our graduate students over 30 years from whom we have learned much and with whom we have argued often, particularly, Selesnick, Gopinath, Soewito, and Vargas. We also owe much to the IEEE Signal Processing Society and to Rice University for environments to learn, teach, create, and collaborate. Much of the results in DSP was supported directly or indirectly by the NSF, most recently NSF grant EEC-0538934 in the Partnerships for Innovation program working with National Instruments, Inc.

We particularly thank Texas Instruments and Prentice Hall for returning the copyrights to me so that part of the material in **DFT/FFT and Convolution Algorithms**[link], **Design of Digital Filters**[link], and “Efficient Fourier Transform and Convolution Algorithms" in **Advanced Topics in Signal Processing**[link] could be included here under the Creative Commons Attribution copyright. I also appreciate IEEE policy that allows parts of my papers to be included here.

A rather long list of references is included to point to more background, to more advanced theory, and to applications. A book of Matlab DSP exercises that could be used with this book has been published through Prentice Hall [link], [link]. Some Matlab programs are included to aid in understanding the design algorithms and to actually design filters. LabView from National Instruments is a very useful tool to both learn with and use in application. All of the material in these notes is being put into “Connexions" [link] which is a modern web-based open-content information system www.cnx.org. Further information is available on our web site at www.dsp.rice.edu with links to other related work. We thank Richard Baraniuk, Don Johnson, Ray Wagner, Daniel Williamson, and Marcia Horton for their help.

This version of the book is a draft and will continue to evolve under Connexions. A companion FFT book is being written and is also available in Connexions and print form. All of these two books are in the repository of Connexions and, therefore, available to anyone free to use, reuse, modify, etc. as long as attribution is given.

C. Sidney Burrus

Houston, Texas

June 2008

Continuous-Time Signals

Signals occur in a wide range of physical phenomenon. They might be human speech, blood pressure variations with time, seismic waves, radar and sonar signals, pictures or images, stress and strain signals in a building structure, stock market prices, a city's population, or temperature across a plate. These signals are often modeled or represented by a real or complex valued mathematical function of one or more variables. For example, speech is modeled by a function representing air pressure varying with time. The function is acting as a mathematical analogy to the speech signal and, therefore, is called an **analog** signal. For these signals, the independent variable is time and it changes continuously so that the term **continuous-time** signal is also used. In our discussion, we talk of the mathematical function as the signal even though it is really a model or representation of the physical signal.

The description of signals in terms of their sinusoidal frequency content has proven to be one of the most powerful tools of continuous and discrete-time signal description, analysis, and processing. For that reason, we will start the discussion of signals with a development of Fourier transform methods. We will first review the continuous-time methods of the Fourier series (FS), the Fourier transform or integral (FT), and the Laplace transform (LT). Next the discrete-time methods will be developed in more detail with the discrete Fourier transform (DFT) applied to finite length signals followed by the discrete-time Fourier transform (DTFT) for infinitely long signals and ending with the Z-transform which allows the powerful tools of complex variable theory to be applied.

More recently, a new tool has been developed for the analysis of signals. Wavelets and wavelet transforms [link], [link], [link], [link], [link] are another more flexible expansion system that also can describe continuous and discrete-time, finite or infinite duration signals. We will very briefly introduce the ideas behind wavelet-based signal analysis.

The Fourier Series

The problem of expanding a finite length signal in a trigonometric series was posed and studied in the late 1700's by renowned mathematicians such as Bernoulli, d'Alembert, Euler, Lagrange, and Gauss. Indeed, what we now call the Fourier series and the formulas for the coefficients were used by Euler in 1780. However, it was the presentation in 1807 and the paper in 1822 by Fourier stating that an arbitrary function could be represented by a series of sines and cosines that brought the problem to everyone's attention and started serious theoretical investigations and practical applications that continue to this day [link], [link], [link], [link], [link], [link]. The theoretical work has been at the center of analysis and the practical applications have been of major significance in virtually every field of quantitative science and technology. For these reasons and others, the Fourier series is worth our serious attention in a study of signal processing.

Definition of the Fourier Series

We assume that the signal $x(t)$ to be analyzed is well described by a real or complex valued function of a real variable t defined over a finite interval $\{0 \leq t \leq T\}$. The trigonometric series expansion of $x(t)$ is given by

Equation:

$$x\left(t\right)=\frac{a(0)}{2}+\sum_{k=1}^{\infty}a\left(k\right)\cos\left(\frac{2\pi}{T}kt\right)+b\left(k\right)\sin\left(\frac{2\pi}{T}kt\right).$$

where $x_k\left(t\right)=cos\left(2\pi kt/T\right)$ and $y_k\left(t\right)=sin\left(2\pi kt/T\right)$ are the basis functions for the expansion. The energy or power in an electrical, mechanical, etc. system is a function of the square of voltage, current, velocity, pressure, etc. For this reason, the natural setting for a representation of signals is the Hilbert space of $L^2\left[0,T\right]$. This modern formulation of the problem is developed in [link], [link]. The sinusoidal basis functions in the trigonometric expansion form a complete orthogonal set in $L^2\left[0,T\right]$. The orthogonality is easily seen from inner products

Equation:

$$\left(\cos\left(\frac{2\pi}{T}kt\right),\ \cos\left(\frac{2\pi}{T}\ell t\right)\right)=\int_0^T\left(\cos\left(\frac{2\pi}{T}kt\right)\ \cos\left(\frac{2\pi}{T}\ell t\right)\right)dt=\delta\left(k-\ell\right)$$

and

Equation:

$$\left(\cos\left(\frac{2\pi}{T}kt\right),\ \sin\left(\frac{2\pi}{T}\ell t\right)\right)=\int_0^T\left(\cos\left(\frac{2\pi}{T}kt\right)\ \sin\left(\frac{2\pi}{T}\ell t\right)\right)dt=0$$

where $\delta(t)$ is the Kronecker delta function with $\delta(0)=1$ and $\delta(k\neq 0)=0$. Because of this, the kth coefficients in the series can be found by taking the inner product of $x(t)$ with the kth basis functions. This gives for the coefficients

Equation:

$$a\left(k\right)=\frac{2}{T}\int_0^T x\left(t\right)\cos\left(\frac{2\pi}{T}kt\right)dt$$

and

Equation:

$$b\left(k\right)=\frac{2}{T}\int_{0}^{T}x\left(t\right)\sin\left(\frac{2\pi}{T}kt\right)dt$$

where T is the time interval of interest or the period of a periodic signal. Because of the orthogonality of the basis functions, a finite Fourier series formed by truncating the infinite series is an optimal least squared error approximation to $x(t)$. If the finite series is defined by

Equation:

$$\widehat{x}\left(t\right)=\frac{a(0)}{2}+\sum_{k=1}^{N}a\left(k\right)\cos\left(\frac{2\pi}{T}kt\right)+b\left(k\right)\sin\left(\frac{2\pi}{T}kt\right),$$

the squared error is

Equation:

$$\varepsilon=\frac{1}{T}\int_{0}^{T}\left|x\left(t\right)-\widehat{x}\left(t\right)\right|^{2}dt$$

which is minimized over all $a(k)$ and $b(k)$ by [link] and [link]. This is an extraordinarily important property.

It follows that if $x\left(t\right)\in L^{2}\left[0,T\right]$, then the series converges to $x(t)$ in the sense that $\varepsilon\to0$ as $N\to\infty$[link], [link]. The question of point-wise convergence is more difficult. A sufficient condition that is adequate for most application states: If $f(x)$ is bounded, is piece-wise continuous, and has no more than a finite number of maxima over an interval, the Fourier series converges point-wise to $f(x)$ at all points of continuity and to the arithmetic mean at points of discontinuities. If $f(x)$ is continuous, the series converges uniformly at all points [link], [link], [link].

A useful condition [link], [link] states that if $x(t)$ and its derivatives through the *q*th derivative are defined and have bounded variation, the Fourier coefficients $a(k)$ and $b(k)$ asymptotically drop off at least as fast as $\frac{1}{k^{q+1}}$ as $k\to\infty$. This ties global rates of convergence of the coefficients to local smoothness conditions of the function.

The form of the Fourier series using both sines and cosines makes determination of the peak value or of the location of a particular frequency term difficult. A different form that explicitly gives the peak value of the sinusoid of that frequency and the location or phase shift of that sinusoid is given by

Equation:

$$x\left(t\right)=\frac{d(0)}{2}+\sum_{k=1}^{\infty}d\left(k\right)\cos\left(\frac{2\pi}{T}kt+\theta\left(k\right)\right)$$

and, using Euler's relation and the usual electrical engineering notation of $j=\sqrt{-1}$,
Equation:

$$e^{jx}=\cos\left(x\right)+j\sin\left(x\right),$$

the complex exponential form is obtained as
Equation:

$$x\left(t\right)=\sum_{k=-\infty}^{\infty}c\left(k\right)e^{j\frac{2\pi}{T}kt}$$

where
Equation:

$$c(k)=a(k)+j\,b(k).$$

The coefficient equation is
Equation:

$$c\left(k\right)=\frac{1}{T}\int_{0}^{T}x\left(t\right)e^{-j\frac{2\pi}{T}kt}dt$$

The coefficients in these three forms are related by
Equation:

$$\left|d\right|^{2}=\left|c\right|^{2}=a^{2}+b^{2}$$

and
Equation:

$$\theta=arg\left\{c\right\}=\tan^{-1}\left(\frac{b}{a}\right)$$

It is easier to evaluate a signal in terms of $c(k)$ or $d(k)$ and $\theta(k)$ than in terms of $a(k)$ and $b(k)$. The first two are polar representation of a complex value and the last is rectangular.

The exponential form is easier to work with mathematically.

Although the function to be expanded is defined only over a specific finite region, the series converges to a function that is defined over the real line and is periodic. It is equal to the original function over the region of definition and is a periodic extension outside of the region. Indeed, one could artificially extend the given function at the outset and then the expansion would converge everywhere.

A Geometric View

It can be very helpful to develop a geometric view of the Fourier series where $x(t)$ is considered to be a vector and the basis functions are the coordinate or basis vectors. The coefficients become the projections of $x(t)$ on the coordinates. The ideas of a measure of distance, size, and orthogonality are important and the definition of error is easy to picture. This is done in [link], [link], [link] using Hilbert space methods.

Properties of the Fourier Series

The properties of the Fourier series are important in applying it to signal analysis and to interpreting it. The main properties are given here using the notation that the Fourier series of a real valued function $x(t)$ over $\{0 \leq t \leq T\}$ is given by $\mathscr{F}\{x(t)\} = c(k)$ and $\widetilde{x}\,(t)$ denotes the periodic extensions of $x(t)$.

1. Linear: $\mathscr{F}\{x + y\} = \mathscr{F}\{x\} + \mathscr{F}\{y\}$
   Idea of superposition. Also scalability: $\mathscr{F}\{ax\} = a\mathscr{F}\{x\}$
2. Extensions of $x(t)$: $\widetilde{x}\,(t) = \widetilde{x}\,(t + T)$
   $\widetilde{x}\,(t)$ is periodic.
3. Even and Odd Parts: $x(t) = u(t) + jv(t)$ and
   $C\,(k) = A\,(k) + jB\,(k) = |C\,(k)|\, e^{j\theta(k)}$

u	v	A	B	$\|C\|$	θ
even	0	even	0	even	0
odd	0	0	odd	even	0

0	even	0	even	even	$\pi/2$
0	odd	odd	0	even	$\pi/2$

4. Convolution: If continuous cyclic convolution is defined by
   Equation:

$$y(t) = h(t) \circ x(t) = \int_0^T \widetilde{h}(t-\tau)\,\widetilde{x}(\tau)\,d\tau$$

   then $\mathscr{F}\{h(t) \circ x(t)\} = \mathscr{F}\{h(t)\}\,\mathscr{F}\{x(t)\}$
5. Multiplication: If discrete convolution is defined by
   Equation:

$$e(n) = d(n)^* c(n) = \sum_{m=-\infty}^{\infty} d(m)\,c(n-m)$$

   then $\mathscr{F}\{h(t)\,x(t)\} = \mathscr{F}\{h(t)\}^*\mathscr{F}\{x(t)\}$
   This property is the inverse of property 4 and vice versa.
6. Parseval: $\frac{1}{T}\int_0^T |x(t)|^2 dt = \sum_{k=-\infty}^{\infty} |C(k)|^2$
   This property says the energy calculated in the time domain is the same as that calculated in the frequency (or Fourier) domain.
7. Shift: $\mathscr{F}\{\widetilde{x}(t-t_0)\} = C(k)\,e^{-j2\pi t_0 k/T}$
   A shift in the time domain results in a linear phase shift in the frequency domain.
8. Modulate: $\mathscr{F}\left\{x(t)\,e^{j2\pi Kt/T}\right\} = C(k-K)$
   Modulation in the time domain results in a shift in the frequency domain. This property is the inverse of property 7.
9. Orthogonality of basis functions:
   Equation:

$$\int_0^T e^{-j2\pi mt/T}\,e^{j2\pi nt/T}\,dt = T\,\delta(n-m) = \begin{cases} T & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

   Orthogonality allows the calculation of coefficients using inner products in [link] and [link]. It also allows Parseval's Theorem in property 6. A relaxed version of orthogonality is called “tight frames" and is important in over-specified systems, especially in wavelets.

Examples

- An example of the Fourier series is the expansion of a square wave signal with period 2π. The expansion is
  Equation:

$$x(t) = \frac{4}{\pi}\left[\sin(t) + \frac{1}{3}\sin(3t) + \frac{1}{5}\sin(5t)\cdots\right].$$

  Because $x(t)$ is odd, there are no cosine terms (all $a(k) = 0$) and, because of its symmetries, there are no even harmonics (even k terms are zero). The function is well defined and bounded; its derivative is not, therefore, the coefficients drop off as $\frac{1}{k}$.
- A second example is a triangle wave of period 2π. This is a continuous function where the square wave was not. The expansion of the triangle wave is
  Equation:

$$x(t) = \frac{4}{\pi}\left[\sin(t) - \frac{1}{3^2}\sin(3t) + \frac{1}{5^2}\sin(5t) + \cdots\right].$$

  Here the coefficients drop off as $\frac{1}{k^2}$ since the function and its first derivative exist and are bounded.

Note the derivative of a triangle wave is a square wave. Examine the series coefficients to see this. There are many books and web sites on the Fourier series that give insight through examples and demos.

Theorems on the Fourier Series

Four of the most important theorems in the theory of Fourier analysis are the inversion theorem, the convolution theorem, the differentiation theorem, and Parseval's theorem [link].

- The inversion theorem is the truth of the transform pair given in [link], [link], and [link]..
- The convolution theorem is property 4.
- The differentiation theorem says that the transform of the derivative of a function is $j\omega$ times the transform of the function.
- Parseval's theorem is given in property 6.

All of these are based on the orthogonality of the basis function of the Fourier series and integral and all require knowledge of the convergence of the sums and integrals. The practical and theoretical use of Fourier analysis is greatly expanded if use is made of distributions or generalized functions (e.g. Dirac delta functions, $\delta(t)$) [link], [link]. Because energy is an important measure of a function in signal processing applications,

the Hilbert space of L^2 functions is a proper setting for the basic theory and a geometric view can be especially useful [link], [link].

The following theorems and results concern the existence and convergence of the Fourier series and the discrete-time Fourier transform [link]. Details, discussions and proofs can be found in the cited references.

- If $f(x)$ has bounded variation in the interval $(-\pi, \pi)$, the Fourier series corresponding to $f(x)$ converges to the value $f(x)$ at any point within the interval, at which the function is continuous; it converges to the value $\frac{1}{2}[f(x+0)+f(x-0)]$ at any such point at which the function is discontinuous. At the points π, $-\pi$ it converges to the value $\frac{1}{2}[f(-\pi+0)+f(\pi-0)]$. [link]
- If $f(x)$ is of bounded variation in $(-\pi, \pi)$, the Fourier series converges to $f(x)$, uniformly in any interval (a, b) in which $f(x)$ is continuous, the continuity at a and b being on both sides. [link]
- If $f(x)$ is of bounded variation in $(-\pi, \pi)$, the Fourier series converges to $\frac{1}{2}[f(x+0)+f(x-0)]$, bounded throughout the interval $(-\pi, \pi)$. [link]
- If $f(x)$ is bounded and if it is continuous in its domain at every point, with the exception of a finite number of points at which it may have ordinary discontinuities, and if the domain may be divided into a finite number of parts, such that in any one of them the function is monotone; or, in other words, the function has only a finite number of maxima and minima in its domain, the Fourier series of $f(x)$ converges to $f(x)$ at points of continuity and to $\frac{1}{2}[f(x+0)+f(x-0)]$ at points of discontinuity. [link], [link]
- If $f(x)$ is such that, when the arbitrarily small neighborhoods of a finite number of points in whose neighborhood $|f(x)|$ has no upper bound have been excluded, $f(x)$ becomes a function with bounded variation, then the Fourier series converges to the value $\frac{1}{2}[f(x+0)+f(x-0)]$, at every point in $(-\pi, \pi)$, except the points of infinite discontinuity of the function, provided the improper integral $\int_{-\pi}^{\pi} f(x)dx$ exist, and is absolutely convergent. [link]
- If f is of bounded variation, the Fourier series of f converges at every point x to the value $[f(x+0)+f(x-0)]/2$. If f is, in addition, continuous at every point of an interval $I=(a, b)$, its Fourier series is uniformly convergent in I. [link]
- If $a(k)$ and $b(k)$ are absolutely summable, the Fourier series converges uniformly to $f(x)$ which is continuous. [link]
- If $a(k)$ and $b(k)$ are square summable, the Fourier series converges to $f(x)$ where it is continuous, but not necessarily uniformly. [link]
- Suppose that $f(x)$ is periodic, of period X, is defined and bounded on $[0, X]$ and that at least one of the following four conditions is satisfied: (i) f is piecewise monotonic on $[0, X]$, (ii) f has a finite number of maxima and minima on $[0, X]$ and a finite number of discontinuities on $[0, X]$, (iii) f is of bounded variation on $[0, X]$, (iv) f is piecewise smooth on $[0, X]$: then it will follow that the Fourier series coefficients

may be defined through the defining integral, using proper Riemann integrals, and that the Fourier series converges to $f(x)$ at a.a.x, to $f(x)$ at each point of continuity of f, and to the value $\frac{1}{2}[f(x^-) + f(x^+)]$ at all x. [link]

- For any $1 \leq p < \infty$ and any $f \in C^p\left(S^1\right)$, the partial sums
  Equation:

$$S_n = S_n(f) = \sum_{|k| \leq n} \widehat{f}(k) e_k$$

  converge to f, uniformly as $n \to \infty$; in fact, $||S_n - f||_\infty$ is bounded by a constant multiple of $n^{-p+1/2}$. [link]

The Fourier series expansion results in transforming a periodic, continuous time function, $\widetilde{x}(t)$, to two discrete indexed frequency functions, $a(k)$ and $b(k)$ that are not periodic.

The Fourier Transform

Many practical problems in signal analysis involve either infinitely long or very long signals where the Fourier series is not appropriate. For these cases, the Fourier transform (FT) and its inverse (IFT) have been developed. This transform has been used with great success in virtually all quantitative areas of science and technology where the concept of frequency is important. While the Fourier series was used before Fourier worked on it, the Fourier transform seems to be his original idea. It can be derived as an extension of the Fourier series by letting the length or period T increase to infinity or the Fourier transform can be independently defined and then the Fourier series shown to be a special case of it. The latter approach is the more general of the two, but the former is more intuitive [link], [link].

Definition of the Fourier Transform

The Fourier transform (FT) of a real-valued (or complex) function of the real-variable t is defined by
Equation:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, dt$$

giving a complex valued function of the real variable ω representing frequency. The inverse Fourier transform (IFT) is given by
Equation:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)\, e^{j\omega t}\, d\omega.$$

Because of the infinite limits on both integrals, the question of convergence is important. There are useful practical signals that do not have Fourier transforms if only classical functions are allowed because of problems with convergence. The use of delta functions (distributions) in both the time and frequency domains allows a much larger class of signals to be represented [link].

Properties of the Fourier Transform

The properties of the Fourier transform are somewhat parallel to those of the Fourier series and are important in applying it to signal analysis and interpreting it. The main properties are given here using the notation that the FT of a real valued function $x(t)$ over all time t is given by $\mathscr{F}\{x\} = X(\omega)$.

1. Linear: $\mathscr{F}\{x + y\} = \mathscr{F}\{x\} + \mathscr{F}\{y\}$
2. Even and Oddness: if $x(t) = u(t) + jv(t)$ and $X(\omega) = A(\omega) + jB(\omega)$ then

u	v	A	B	$\lvert X\rvert$	θ
even	0	even	0	even	0
odd	0	0	odd	even	0
0	even	0	even	even	$\pi/2$
0	odd	odd	0	even	$\pi/2$

3. Convolution: If continuous convolution is defined by:
   Equation:

$$y(t) = h(t)^{*}x(t) = \int_{-\infty}^{\infty} h(t-\tau)x(\tau)d\tau = \int_{-\infty}^{\infty} h(\lambda)x(t-\lambda)d\lambda$$

   then $\mathscr{F}\{h(t)^{*}x(t)\} = \mathscr{F}\{h(t)\}\mathscr{F}\{x(t)\}$

4. Multiplication: $\mathscr{F}\{h(t)x(t)\} = \frac{1}{2\pi}\mathscr{F}\{h(t)\}*\mathscr{F}\{x(t)\}$
5. Parseval: $\int_{-\infty}^{\infty}|x(t)|^2 dt = \frac{1}{2\pi}\int_{-\infty}^{\infty}|X(\omega)|^2 d\omega$
6. Shift: $\mathscr{F}\{x(t-T)\} = X(\omega)e^{-j\omega T}$
7. Modulate: $\mathscr{F}\{x(t)e^{j2\pi Kt}\} = X(\omega - 2\pi K)$
8. Derivative: $\mathscr{F}\left\{\frac{dx}{dt}\right\} = j\omega X(\omega)$
9. Stretch: $\mathscr{F}\{x(at)\} = \frac{1}{|a|}X(\omega/a)$
10. Orthogonality: $\int_{-\infty}^{\infty} e^{-j\omega_1 t}e^{j\omega_2 t} = 2\pi\delta(\omega_1 - \omega_2)$

Examples of the Fourier Transform

Deriving a few basic transforms and using the properties allows a large class of signals to be easily studied. Examples of modulation, sampling, and others will be given.

- If $x(t) = \delta(t)$ then $X(\omega) = 1$
- If $x(t) = 1$ then $X(\omega) = 2\pi\delta(\omega)$
- If $x(t)$ is an infinite sequence of delta functions spaced T apart, $x(t) = \sum_{n=-\infty}^{\infty}\delta(t - nT)$, its transform is also an infinite sequence of delta functions of weight $2\pi/T$ spaced $2\pi/T$ apart, $X(\omega) = 2\pi\sum_{k=-\infty}^{\infty}\delta(\omega - 2\pi k/T)$.
- Other interesting and illustrative examples can be found in [link], [link].

Note the Fourier transform takes a function of continuous time into a function of continuous frequency, neither function being periodic. If “distribution" or “delta functions" are allowed, the Fourier transform of a periodic function will be a infinitely long string of delta functions with weights that are the Fourier series coefficients.

The Laplace Transform

The Laplace transform can be thought of as a generalization of the Fourier transform in order to include a larger class of functions, to allow the use of complex variable theory, to solve initial value differential equations, and to give a tool for input-output description of linear systems. Its use in system and signal analysis became popular in the 1950's and remains as the central tool for much of continuous time system theory. The question of convergence becomes still more complicated and depends on complex values of s used in the inverse transform which must be in a “region of convergence" (ROC).

Definition of the Laplace Transform

The definition of the Laplace transform (LT) of a real valued function defined over all positive time t is

Equation:

$$F(s) = \int_{-\infty}^{\infty} f(t)\, e^{-st}\, dt$$

and the inverse transform (ILT) is given by the complex contour integral
Equation:

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s)\, e^{st}\, ds$$

where $s = \sigma + j\omega$ is a complex variable and the path of integration for the ILT must be in the region of the s plane where the Laplace transform integral converges. This definition is often called the bilateral Laplace transform to distinguish it from the unilateral transform (ULT) which is defined with zero as the lower limit of the forward transform integral [link]. Unless stated otherwise, we will be using the bilateral transform.

Notice that the Laplace transform becomes the Fourier transform on the imaginary axis, for $s = j\omega$. If the ROC includes the $j\omega$ axis, the Fourier transform exists but if it does not, only the Laplace transform of the function exists.

There is a considerable literature on the Laplace transform and its use in continuous-time system theory. We will develop most of these ideas for the discrete-time system in terms of the z-transform later in this chapter and will only briefly consider only the more important properties here.

The unilateral Laplace transform cannot be used if useful parts of the signal exists for negative time. It does not reduce to the Fourier transform for signals that exist for negative time, but if the negative time part of a signal can be neglected, the unilateral transform will converge for a much larger class of function that the bilateral transform will. It also makes the solution of linear, constant coefficient differential equations with initial conditions much easier.

Properties of the Laplace Transform

Many of the properties of the Laplace transform are similar to those for Fourier transform [link], [link], however, the basis functions for the Laplace transform are not orthogonal. Some of the more important ones are:

1. Linear: $\mathscr{L}\{x + y\} = \mathscr{L}\{x\} + \mathscr{L}\{y\}$

2. Convolution: If $y(t) = h(t)^* x(t) = \int h(t-\tau)\, x(\tau)\, d\tau$
   then $\mathscr{L}\{h(t)^* x(t)\} = \mathscr{L}\{h(t)\}\,\mathscr{L}\{x(t)\}$
3. Derivative: $\mathscr{L}\left\{\frac{dx}{dt}\right\} = s\mathscr{L}\{x(t)\}$
4. Derivative (ULT): $\mathscr{L}\left\{\frac{dx}{dt}\right\} = s\mathscr{L}\{x(t)\} - x(0)$
5. Integral: $\mathscr{L}\left\{\int x(t)\, dt\right\} = \frac{1}{s}\mathscr{L}\{x(t)\}$
6. Shift: $\mathscr{L}\{x(t-T)\} = C(k)\, e^{-Ts}$
7. Modulate: $\mathscr{L}\left\{x(t)\, e^{j\omega_0 t}\right\} = X(s - j\omega_0)$

Examples can be found in [link], [link] and are similar to those of the z-transform presented later in these notes. Indeed, note the parallals and differences in the Fourier series, Fourier transform, and Z-transform.

Discrete-Time Signals

Although the discrete-time signal $x(n)$ could be any ordered sequence of numbers, they are usually samples of a continuous-time signal. In this case, the real or imaginary valued mathematical function $x(n)$ of the integer n is not used as an analogy of a physical signal, but as some representation of it (such as samples). In some cases, the term **digital** signal is used interchangeably with discrete-time signal, or the label digital signal may be use if the function is not real valued but takes values consistent with some hardware system.

Indeed, our very use of the term “discrete-time" indicates the probable origin of the signals when, in fact, the independent variable could be length or any other variable or simply an ordering index. The term “digital" indicates the signal is probably going to be created, processed, or stored using digital hardware. As in the continuous-time case, the Fourier transform will again be our primary tool [link], [link], [link].

Notation has been an important element in mathematics. In some cases, discrete-time signals are best denoted as a sequence of values, in other cases, a vector is created with elements which are the sequence values. In still other cases, a polynomial is formed with the sequence values as coefficients for a complex variable. The vector formulation allows the use of linear algebra and the polynomial formulation allows the use of complex variable theory.

The Discrete Fourier Transform

The description of signals in terms of their sinusoidal frequency content has proven to be as powerful and informative for discrete-time signals as it has for continuous-time signals. It is also probably the most powerful computational tool we will use. We now develop the basic discrete-time methods starting with the discrete Fourier transform (DFT) applied to finite length signals, followed by the discrete-time Fourier transform (DTFT) for infinitely long signals, and ending with the z-transform which uses the powerful tools of complex variable theory.

Definition of the DFT

It is assumed that the signal $x(n)$ to be analyzed is a sequence of N real or complex values which are a function of the integer variable n. The DFT of $x(n)$, also called the spectrum of $x(n)$, is a length N sequence of complex numbers denoted $C(k)$ and defined by
Equation:

$$C\left(k\right) \;=\; \sum_{n=0}^{N-1} x\left(n\right) e^{-j\frac{2\pi}{N}nk}$$

using the usual engineering notation: $j = \sqrt{-1}$. The inverse transform (IDFT) which retrieves $x(n)$ from $C(k)$ is given by
Equation:

$$x\left(n\right) \;=\; \frac{1}{N}\sum_{k=0}^{N-1} C\left(k\right) e^{j\frac{2\pi}{N}nk}$$

which is easily verified by substitution into [link]. Indeed, this verification will require using the orthogonality of the basis function of the DFT which is
Equation:

$$\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}mk} e^{j\frac{2\pi}{N}nk} \;=\; \begin{cases} N & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

The exponential basis functions, $e^{-j\frac{2\pi}{N}k}$, for $k \in \{0, N-1\}$, are the N values of the Nth roots of unity (the N zeros of the polynomial $(s-1)^N$). This property is what connects the DFT to convolution and allows efficient algorithms for calculation to be developed [link]. They are used so often that the following notation is defined by
Equation:

$$W_N \;=\; e^{-j\frac{2\pi}{N}}$$

with the subscript being omitted if the sequence length is obvious from context. Using this notation, the DFT becomes
Equation:

$$C\left(k\right) \;=\; \sum_{n=0}^{N-1} x\left(n\right) W_N^{nk}$$

One should notice that with the finite summation of the DFT, there is no question of convergence or of the ability to interchange the order of summation. No “delta functions” are needed and the N transform values can be calculated exactly (within the

accuracy of the computer or calculator used) from the N signal values with a finite number of arithmetic operations.

Matrix Formulation of the DFT

There are several advantages to using a matrix formulation of the DFT. This is given by writing [link] or [link] in matrix operator form as
Equation:

$$\begin{matrix} C_0 \\ C_1 \\ C_2 \\ \vdots \\ C_{N-1} \end{matrix} = \begin{matrix} W^0 & W^0 & W^0 & \cdots & W^0 \\ W^0 & W^1 & W^2 & & \\ W^0 & W^2 & W^4 & & \\ \vdots & & & & \vdots \\ W^0 & & \cdots & & W^{(N-1)(N-1)} \end{matrix} \begin{matrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{matrix}$$

or
Equation:

$$\mathbf{C} = \mathbf{F}\mathbf{x}.$$

The orthogonality of the basis function in [link] shows up in this matrix formulation by the columns of F being orthogonal to each other as are the rows. This means that $\mathbf{F^T F} = k\mathbf{I}$, where k is a scalar constant, and, therefore, $\mathbf{F^T} = k\mathbf{F^{-1}}$. This is called a unitary operator.

The definition of the DFT in [link] emphasizes the fact that each of the N DFT values are the sum of N products. The matrix formulation in [link] has two interpretations. Each k-th DFT term is the inner product of two vectors, k-th row of $\mathbf{F}$ and $\mathbf{x}$; or, the DFT vector, $\mathbf{C}$ is a weighted sum of the N columns of $\mathbf{F}$ with weights being the elements of the signal vector $\mathbf{x}$. A third view of the DFT is the operator view which is simply the single matrix equation [link].

It is instructive at this point to write a computer program to calculate the DFT of a signal. In Matlab [link], there is a pre-programmed function to calculate the DFT, but that hides the scalar operations. One should program the transform in the scalar interpretive language of Matlab or some other lower level language such as FORTRAN, C, BASIC, Pascal, etc. This will illustrate how many multiplications and

additions and trigonometric evaluations are required and how much memory is needed. Do not use a complex data type which also hides arithmetic, but use Euler's relations
Equation:

$$e^{jx} = \cos(x) + j\sin(x)$$

to explicitly calculate the real and imaginary part of $C(k)$.

If Matlab is available, first program the DFT using only scalar operations. It will require two nested loops and will run rather slowly because the execution of loops is interpreted. Next, program it using vector inner products to calculate each $C(k)$ which will require only one loop and will run faster. Finally, program it using a single matrix multiplication requiring no loops and running much faster. Check the memory requirements of the three approaches.

The DFT and IDFT are a completely well-defined, legitimate transform pair with a sound theoretical basis that do not need to be derived from or interpreted as an approximation to the continuous-time Fourier series or integral. The discrete-time and continuous-time transforms and other tools are related and have parallel properties, but neither depends on the other.

The notation used here is consistent with most of the literature and with the standards given in [link]. The independent index variable n of the signal $x(n)$ is an integer, but it is usually interpreted as time or, occasionally, as distance. The independent index variable k of the DFT $C(k)$ is also an integer, but it is generally considered as frequency. The DFT is called the spectrum of the signal and the magnitude of the complex valued DFT is called the magnitude of that spectrum and the angle or argument is called the phase.

Extensions of x(n)

Although the finite length signal $x(n)$ is defined only over the interval $\{0 \leq n \leq (N-1)\}$, the IDFT of $C(k)$ can be evaluated outside this interval to give well defined values. Indeed, this process gives the periodic property 4. There are two ways of formulating this phenomenon. One is to periodically extend $x(n)$ to $-\infty$ and $+\infty$ and work with this new signal. A second more general way is evaluate all indices n and k modulo N. Rather than considering the periodic extension of $x(n)$ on the line of integers, the finite length line is formed into a circle or a line around a cylinder so that after counting to $N-1$, the next number is zero, not a periodic replication of it. The periodic extension is easier to visualize initially and is more commonly used for

the definition of the DFT, but the evaluation of the indices by residue reduction modulo N is a more general definition and can be better utilized to develop efficient algorithms for calculating the DFT [link].

Since the indices are evaluated only over the basic interval, any values could be assigned $x(n)$ outside that interval. The periodic extension is the choice most consistent with the other properties of the transform, however, it could be assigned to zero [link]. An interesting possibility is to artificially create a length $2N$ sequence by appending $x(-n)$ to the end of $x(n)$. This would remove the discontinuities of periodic extensions of this new length $2N$ signal and perhaps give a more accurate measure of the frequency content of the signal with no artifacts caused by “end effects". Indeed, this modification of the DFT gives what is called the discrete cosine transform (DCT) [link]. We will assume the implicit periodic extensions to $x(n)$ with no special notation unless this characteristic is important, then we will use the notation $\widetilde{x}\,(n)$.

Convolution

Convolution is an important operation in signal processing that is in some ways more complicated in discrete-time signal processing than in continuous-time signal processing and in other ways easier. The basic input-output relation for a discrete-time system is given by so-called linear or non-cyclic convolution defined and denoted by
Equation:

$$y\,(n) \;=\; \sum_{m=-\infty}^{\infty} h\,(m)\,x\,(n-m) \;=\; h\,(n)*x\,(n)$$

where $x(n)$ is the perhaps infinitely long input discrete-time signal, $h(n)$ is the perhaps infinitely long impulse response of the system, and $y(n)$ is the output. The DFT is, however, intimately related to cyclic convolution, not non-cyclic convolution. Cyclic convolution is defined and denoted by
Equation:

$$\tilde{y}\,(n) \;=\; \sum_{m=0}^{N-1} \widetilde{h}\,(m)\,\widetilde{x}\,(n-m) \;=\; h\,(n)\circ x\,(n)$$

where either all of the indices or independent integer variables are evaluated modulo N or all of the signals are periodically extended outside their length N domains.

This cyclic (sometimes called circular) convolution can be expressed as a matrix operation by converting the signal $h(n)$ into a matrix operator as
Equation:

$$\mathbf{H} = \begin{matrix} h_0 & h_{L-1} & h_{L-2} & \cdots & h_1 \\ h_1 & h_0 & h_{L-1} & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \\ h_{L-1} & & \cdots & & h_0 \end{matrix},$$

The cyclic convolution can then be written in matrix notation as
Equation:

$$\mathbf{Y} = \mathbf{HX}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are column matrices or vectors of the input and output values respectively.

Because non-cyclic convolution is often what you want to do and cyclic convolution is what is related to the powerful DFT, we want to develop a way of doing non-cyclic convolution by doing cyclic convolution.

The convolution of a length N sequence with a length M sequence yields a length $N + M - 1$ output sequence. The calculation of non-cyclic convolution by using cyclic convolution requires modifying the signals by appending zeros to them. This will be developed later.

Properties of the DFT

The properties of the DFT are extremely important in applying it to signal analysis and to interpreting it. The main properties are given here using the notation that the DFT of a length-N complex sequence $x(n)$ is $\mathscr{F}\{x(n)\} = C(k)$.

1. Linear Operator: $\mathscr{F}\{x(n) + y(n)\} = \mathscr{F}\{x(n)\} + \mathscr{F}\{y(n)\}$
2. Unitary Operator: $\mathbf{F^{-1}} = \frac{1}{N}\mathbf{F^T}$
3. Periodic Spectrum: $C(k) = C(k + N)$
4. Periodic Extensions of $x(n)$: $x(n) = x(n + N)$

5. Properties of Even and Odd Parts: $x(n) = u(n) + jv(n)$ and $C(k) = A(k) + jB(k)$

u	v	A	B	$\|C\|$	θ
even	0	even	0	even	0
odd	0	0	odd	even	$\pi/2$
0	even	0	even	even	$\pi/2$
0	odd	odd	0	even	0

6. Cyclic Convolution: $\mathscr{F}\{h(n) \circ x(n)\} = \mathscr{F}\{h(n)\}\mathscr{F}\{x(n)\}$
7. Multiplication: $\mathscr{F}\{h(n)x(n)\} = \mathscr{F}\{h(n)\} \circ \mathscr{F}\{x(n)\}$
8. Parseval: $\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N}\sum_{k=0}^{N-1} |C(k)|^2$
9. Shift: $\mathscr{F}\{x(n-M)\} = C(k)e^{-j2\pi Mk/N}$
10. Modulate: $\mathscr{F}\left\{x(n)e^{j2\pi Kn/N}\right\} = C(k-K)$
11. Down Sample or Decimate: $\mathscr{F}\{x(Kn)\} = \frac{1}{K}\sum_{m=0}^{K-1} C(k+Lm)$ where $N = LK$
12. Up Sample or Stretch: If $x_s(2n) = x(n)$ for integer n and zero otherwise, then $\mathscr{F}\{x_s(n)\} = C(k)$, for $k = 0, 1, 2, \ldots, 2N-1$
13. N Roots of Unity: $\left(W_N^k\right)^N = 1$ for $k = 0, 1, 2, \ldots, N-1$
14. Orthogonality:
    Equation:

$$\sum_{k=0}^{N-1} e^{-j2\pi mk/N} e^{j2\pi nk/N} = \begin{cases} N & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$$

15. Diagonalization of Convolution: If cyclic convolution is expressed as a matrix operation by $\mathbf{y} = \mathbf{Hx}$ with $\mathbf{H}$ given by [link], the DFT operator diagonalizes the convolution operator $\mathbf{H}$, or $\mathbf{F^T HF} = \mathbf{H_d}$ where $\mathbf{H_d}$ is a diagonal matrix with the N values of the DFT of $h(n)$ on the diagonal. This is a matrix statement of Property 6. Note the columns of $\mathbf{F}$ are the N eigenvectors of $\mathbf{H}$, independent of the values of $h(n)$.

One can show that any “kernel" of a transform that would support cyclic, length-N convolution must be the N roots of unity. This says the DFT is the only transform over the complex number field that will support convolution. However, if one considers various finite fields or rings, an interesting transform, called the Number Theoretic Transform, can be defined and used because the roots of unity are simply two raised to a powers which is a simple word shift for certain binary number representations [link], [link].

Examples of the DFT

It is very important to develop insight and intuition into the DFT or spectral characteristics of various standard signals. A few DFT's of standard signals together with the above properties will give a fairly large set of results. They will also aid in quickly obtaining the DFT of new signals. The discrete-time impulse $\delta(n)$ is defined by
Equation:

$$\delta(n) = \begin{cases} 1 & \text{when } n = 0 \\ 0 & \text{otherwise} \end{cases}$$

The discrete-time pulse $\sqcap_M(n)$ is defined by
Equation:

$$\sqcap_M(n) = \begin{cases} 1 & \text{when } n = 0, 1, \cdots, M-1 \\ 0 & \text{otherwise} \end{cases}$$

Several examples are:

- $DFT\{\delta(n)\} = 1$, The DFT of an impulse is a constant.
- $DFT\{1\} = N\delta(k)$, The DFT of a constant is an impulse.
- $DFT\left\{e^{j2\pi Kn/N}\right\} = N\delta(k-K)$
- $DFT\{\cos(2\pi Mn/N) = \frac{N}{2}[\delta(k-M) + \delta(k+M)]$
- $DFT\{\sqcap_M(n)\} = \frac{\sin\left(\frac{\pi}{N}Mk\right)}{\sin\left(\frac{\pi}{N}k\right)}$

These examples together with the properties can generate a still larger set of interesting and enlightening examples. Matlab can be used to experiment with these results and to gain insight and intuition.

The Discrete-Time Fourier Transform

In addition to finite length signals, there are many practical problems where we must be able to analyze and process essentially infinitely long sequences. For continuous-time signals, the Fourier series is used for finite length signals and the Fourier transform or integral is used for infinitely long signals. For discrete-time signals, we have the DFT for finite length signals and we now present the discrete-time Fourier transform (DTFT) for infinitely long signals or signals that are longer than we want to specify [link]. The DTFT can be developed as an extension of the DFT as N goes to infinity or the DTFT can be independently defined and then the DFT shown to be a special case of it. We will do the latter.

Definition of the DTFT

The DTFT of a possibly infinitely long real (or complex) valued sequence $f(n)$ is defined to be
Equation:

$$F(\omega) = \sum_{-\infty}^{\infty} f(n) e^{-j\omega n}$$

and its inverse denoted IDTFT is given by
Equation:

$$f(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(\omega) e^{j\omega n} d\omega.$$

Verification by substitution is more difficult than for the DFT. Here convergence and the interchange of order of the sum and integral are serious questions and have been the topics of research over many years. Discussions of the Fourier transform and series for engineering applications can be found in [link], [link]. It is necessary to allow distributions or delta functions to be used to gain the full benefit of the Fourier transform.

Note that the definition of the DTFT and IDTFT are the same as the definition of the IFS and FS respectively. Since the DTFT is a continuous periodic function of ω, its Fourier series is a discrete set of values which turn out to be the original signal. This duality can be helpful in developing properties and gaining insight into various problems. The conditions on a function to determine if it can be expanded in a FS are

exactly the conditions on a desired frequency response or spectrum that will determine if a signal exists to realize or approximate it.

Properties

The properties of the DTFT are similar to those for the DFT and are important in the analysis and interpretation of long signals. The main properties are given here using the notation that the DTFT of a complex sequence $x(n)$ is $\mathscr{F}\{x(n)\} \; = \; X(\omega)$.

1. Linear Operator: $\mathscr{F}\{x+y\} \; = \; \mathscr{F}\{x\}+\mathscr{F}\{y\}$
2. Periodic Spectrum: $X(\omega) \; = \; X(\omega+2\pi)$
3. Properties of Even and Odd Parts: $x(n) \; = \; u(n)+jv(n)$ and $X(\omega) \; = \; A(\omega)+jB(\omega)$

u	v	A	B	$\lvert X\rvert$	θ
even	0	even	0	even	0
odd	0	0	odd	even	0
0	even	0	even	even	$\pi/2$
0	odd	odd	0	even	$\pi/2$

4. Convolution: If non-cyclic or linear convolution is defined by: $y\left(n\right) \; = \; h\left(n\right)^{*}x\left(n\right) \; = \; \sum_{m=-\infty}^{\infty}h\left(n-m\right)x\left(m\right) \; = \; \sum_{k=-\infty}^{\infty}h\left(k\right)x\left(n-k\right)$ then $\mathscr{F}\{h(n)^{*}x(n)\} \; = \; \mathscr{F}\{h(n)\}\mathscr{F}\{x(n)\}$
5. Multiplication: If cyclic convolution is defined by: $Y\left(\omega\right) \; = \; H\left(\omega\right)\circ X\left(\omega\right) \; = \; \int_{0}^{T}\widetilde{H}\left(\omega-\Omega\right)\widetilde{X}\left(\Omega\right)d\Omega$ $\mathscr{F}\left\{h\left(n\right)x\left(n\right)\right\} \; = \; \frac{1}{2\pi}\mathscr{F}\left\{h\left(n\right)\right\}\circ\mathscr{F}\left\{x\left(n\right)\right\}$
6. Parseval: $\sum_{n=-\infty}^{\infty}\left|x\left(n\right)\right|^{2} \; = \; \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|X\left(\omega\right)\right|^{2}d\omega$
7. Shift: $\mathscr{F}\left\{x\left(n-M\right)\right\} \; = \; X\left(\omega\right)e^{-j\omega M}$
8. Modulate: $\mathscr{F}\left\{x\left(n\right)e^{j\omega_{0}n}\right\} \; = \; X\left(\omega-\omega_{0}\right)$
9. Sample: $\mathscr{F}\left\{x\left(Kn\right)\right\} \; = \; \frac{1}{K}\sum_{m=0}^{K-1}X\left(\omega+Lm\right)$ where $N \; = \; LK$

10. Stretch: $\mathscr{F}\{x_s(n)\} = X(\omega)$, for $-K\pi \leq \omega \leq K\pi$ where $x_s(Kn) = x(n)$ for integer n and zero otherwise.
11. Orthogonality: $\sum_{n=-\infty}^{\infty} e^{-j\omega_1 n} e^{-j\omega_2 n} = 2\pi\delta(\omega_1 - \omega_2)$

Evaluation of the DTFT by the DFT

If the DTFT of a finite sequence is taken, the result is a continuous function of ω. If the DFT of the same sequence is taken, the results are N evenly spaced samples of the DTFT. In other words, the DTFT of a finite signal can be evaluated at N points with the DFT.
Equation:

$$X(\omega) = DTFT\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$

and because of the finite length
Equation:

$$X(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}.$$

If we evaluate ω at N equally space points, this becomes
Equation:

$$X\left(\frac{2\pi}{N}k\right) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn}$$

which is the DFT of $x(n)$. By adding zeros to the end of $x(n)$ and taking a longer DFT, any density of points can be evaluated. This is useful in interpolation and in plotting the spectrum of a finite length signal. This is discussed further in Sampling, Up-Sampling, Down-Sampling, and Multi-Rate Processing.

There is an interesting variation of the Parseval's theorem for the DTFT of a finite length-N signal. If $x(n) \neq 0$ for $0 \geq n \geq N-1$, and if $L \geq N$, then
Equation:

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{L} \sum_{k=0}^{L-1} |X(2\pi k/L)|^2 = \frac{1}{\pi} \int_0^{\pi} |X(\omega)|^2 \, d\omega.$$

The second term in [link] says the Riemann sum is equal to its limit in this case.

Examples of DTFT

As was true for the DFT, insight and intuition is developed by understanding the properties and a few examples of the DTFT. Several examples are given below and more can be found in the literature [link], [link], [link]. Remember that while in the case of the DFT signals were defined on the region $\{0 \leq n \leq (N-1)\}$ and values outside that region were periodic extensions, here the signals are defined over all integers and are not periodic unless explicitly stated. The spectrum is periodic with period 2π.

- $DTFT\{\delta(n)\} = 1$ for all frequencies.
- **Equation:**

$$DTFT\{1\} = 2\pi\delta(\omega)$$

- **Equation:**

$$DTFT\left\{e^{j\omega_0 n}\right\} = 2\pi\delta(\omega - \omega_0)$$

- **Equation:**

$$DTFT\left\{\cos(\omega_0 n)\right\} = \pi\left[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)\right]$$

- **Equation:**

$$DTFT\left\{\sqcap_M(n)\right\} = \frac{\sin(\omega M k/2)}{\sin(\omega k/2)}$$

The Z-Transform

The z-transform is an extension of the DTFT in a way that is analogous to the Laplace transform for continuous-time signals being an extension of the Fourier transform. It allows the use of complex variable theory and is particularly useful in analyzing and describing systems. The question of convergence becomes still more complicated and

depends on values of z used in the inverse transform which must be in the “region of convergence" (ROC).

Definition of the Z-Transform

The z-transform (ZT) is defined as a polynomial in the complex variable z with the discrete-time signal values as its coefficients [link], [link], [link]. It is given by
Equation:

$$F(z) = \sum_{n=-\infty}^{\infty} f(n) z^{-n}$$

and the inverse transform (IZT) is
Equation:

$$f(n) = \frac{1}{2\pi j} \oint_{ROC} F(z) z^{n-1} dz.$$

The inverse transform can be derived by using the residue theorem [link], [link] from complex variable theory to find $f(0)$ from $z^{-1}F(z)$, $f(1)$ from $F(z)$, $f(2)$ from $zF(z)$, and in general, $f(n)$ from $z^{n-1}F(z)$. Verification by substitution is more difficult than for the DFT or DTFT. Here convergence and the interchange of order of the sum and integral is a serious question that involves values of the complex variable z. The complex contour integral in [link] must be taken in the ROC of the z plane.

A unilateral z-transform is sometimes needed where the definition [link] uses a lower limit on the transform summation of zero. This allow the transformation to converge for some functions where the regular bilateral transform does not, it provides a straightforward way to solve initial condition difference equation problems, and it simplifies the question of finding the ROC. The bilateral z-transform is used more for signal analysis and the unilateral transform is used more for system description and analysis. Unless stated otherwise, we will be using the bilateral z-transform.

Properties

The properties of the ZT are similar to those for the DTFT and DFT and are important in the analysis and interpretation of long signals and in the analysis and description of

discrete-time systems. The main properties are given here using the notation that the ZT of a complex sequence $x(n)$ is $\mathscr{Z}\{x(n)\} = X(z)$.

1. Linear Operator: $\mathscr{Z}\{x+y\} = \mathscr{Z}\{x\} + \mathscr{Z}\{y\}$
2. Relationship of ZT to DTFT: $\mathscr{Z}\{x\}|_{z=e^{j\omega}} = \mathscr{DTFT}\{x\}$
3. Periodic Spectrum: $X\left(e^{j\omega}\right) = X\left(e^{j\omega+2\pi}\right)$
4. Properties of Even and Odd Parts: $x(n) = u(n) + jv(n)$ and $X\left(e^{j\omega}\right) = A\left(e^{j\omega}\right) + jB\left(e^{j\omega}\right)$
   Equation:

u	v	A	B
$even$	0	$even$	0
odd	0	0	odd
0	$even$	0	$even$
0	odd	odd	0

5. Convolution: If discrete non-cyclic convolution is defined by $y(n) = h(n)^* x(n) = \sum_{m=-\infty}^{\infty} h(n-m)x(m) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$ then $\mathscr{Z}\{h(n)^* x(n)\} = \mathscr{Z}\{h(n)\}\mathscr{Z}\{x(n)\}$
6. Shift: $\mathscr{Z}\{x(n+M)\} = z^M X(z)$
7. Shift (unilateral): $\mathscr{Z}\{x(n+m)\} = z^m X(z) - z^m x(0) - z^{m-1}x(1) - \cdots - zx(m-1)$
8. Shift (unilateral): $\mathscr{Z}\{x(n-m)\} = z^{-m} X(z) - z^{-m+1}x(-1) - \cdots - x(-m)$
9. Modulate: $\mathscr{Z}\{x(n)a^n\} = X(z/a)$
10. Time mult.: $\mathscr{Z}\{n^m x(n)\} = (-z)^m \frac{d^m X(z)}{dz^m}$
11. Evaluation: The ZT can be evaluated on the unit circle in the z-plane by taking the DTFT of $x(n)$ and if the signal is finite in length, this can be evaluated at sample points by the DFT.

Examples of the Z-Transform

A few examples together with the above properties will enable one to solve and understand a wide variety of problems. These use the unit step function to remove the negative time part of the signal. This function is defined as
Equation:

$$u\left(n\right) \;=\; \begin{cases} 1 & \text{if } n \geq 0 \\ 0 & \text{if } n < 0 \end{cases}$$

and several bilateral z-transforms are given by

- $\mathscr{Z}\{\delta(n)\} \;=\; 1$ for all z.
- $\mathscr{Z}\left\{u\left(n\right)\right\} \;=\; \frac{z}{z-1}$ for $|z| > 1$.
- $\mathscr{Z}\left\{u\left(n\right)a^n\right\} \;=\; \frac{z}{z-a}$ for $|z| > |a|$.

Notice that these are similar to but not the same as a term of a partial fraction expansion.

Inversion of the Z-Transform

The z-transform can be inverted in three ways. The first two have similar procedures with Laplace transformations and the third has no counter part.

- The z-transform can be inverted by the defined contour integral in the ROC of the complex z plane. This integral can be evaluated using the residue theorem [link], [link].
- The z-transform can be inverted by expanding $\frac{1}{z}F\left(z\right)$ in a partial fraction expansion followed by use of tables for the first or second order terms.
- The third method is not analytical but numerical. If $F\left(z\right) \;=\; \frac{P(z)}{Q(z)}$, $f(n)$ can be obtained as the coefficients of long division.

For example
Equation:

$$\frac{z}{z-a} \;=\; 1 + a\,z^{-1} + a^2 z^{-2} + \cdots$$

which is $u\left(n\right)a^n$ as used in the examples above.

We must understand the role of the ROC in the convergence and inversion of the z-transform. We must also see the difference between the one-sided and two-sided transform.

Solution of Difference Equations using the Z-Transform

The z-transform can be used to convert a difference equation into an algebraic equation in the same manner that the Laplace converts a differential equation in to an algebraic equation. The one-sided transform is particularly well suited for solving initial condition problems. The two unilateral shift properties explicitly use the initial values of the unknown variable.

A difference equation DE contains the unknown function $x(n)$ and shifted versions of it such as $x(n-1)$ or $x(n+3)$. The solution of the equation is the determination of $x(t)$. A linear DE has only simple linear combinations of $x(n)$ and its shifts. An example of a linear second order DE is
Equation:

$$a\,x(n) + b\,x(n-1) + c\,x(n-2) \;=\; f(n)$$

A time invariant or index invariant DE requires the coefficients not be a function of n and the linearity requires that they not be a function of $x(n)$. Therefore, the coefficients are constants.

This equation can be analyzed using classical methods completely analogous to those used with differential equations. A solution of the form $x\,(n) = K\lambda^n$ is substituted into the homogeneous difference equation resulting in a second order characteristic equation whose two roots give a solution of the form $x_h\,(n) \;=\; K_1\lambda_1^n + K_2\lambda_2^n$. A particular solution of a form determined by $f(n)$ is found by the method of undetermined coefficients, convolution or some other means. The total solution is the particular solution plus the solution of the homogeneous equation and the three unknown constants K_i are determined from three initial conditions on $x(n)$.

It is possible to solve this difference equation using z-transforms in a similar way to the solving of a differential equation by use of the Laplace transform. The z-transform converts the difference equation into an algebraic equation. Taking the ZT of both sides of the DE gives
Equation:

$$a\,X\,(z) + b\left[z^{-1}X\,(z) + x\,(-1)\right] + c\left[z^{-2}X\,(z) + z^{-1}x\,(-1) + x\,(-2)\right] \;=\; Y\,(z)$$

solving for $X(z)$ gives
Equation:

$$X(z) \;=\; \frac{z^2\left[Y(z) - b\,x(-1) - x(-2)\right] - z\,c\,x(-1)}{a\,z^2 + b\,z + c}$$

and inversion of this transform gives the solution $x(n)$. Notice that two initial values were required to give a unique solution just as the classical method needed two values.

These are very general methods. To solve an nth order DE requires only factoring an n th order polynomial and performing a partial fraction expansion, jobs that computers are well suited to. There are problems that crop up if the denominator polynomial has repeated roots or if the transform of $y(n)$ has a root that is the same as the homogeneous equation, but those can be handled with slight modifications giving solutions with terms of the from $n\lambda^n$ just as similar problems gave solutions for differential equations of the form $t\,e^{st}$.

The original DE could be rewritten in a different form by shifting the index to give
Equation:

$$a\,x(n+2) + b\,x(n+1) + c\,x(n) \;=\; f(n+2)$$

which can be solved using the second form of the unilateral z-transform shift property.

Region of Convergence for the Z-Transform

Since the inversion integral must be taken in the ROC of the transform, it is necessary to understand how this region is determined and what it means even if the inversion is done by partial fraction expansion or long division. Since all signals created by linear constant coefficient difference equations are sums of geometric sequences (or samples of exponentials), an analysis of these cases will cover most practical situations. Consider a geometric sequence starting at zero.
Equation:

$$f(n) \;=\; u(n)\,a^n$$

with a z-transform
Equation:

$$F(z) \;=\; 1 + a\,z^{-1} + a^2\,z^{-2} + a^3\,z^{-3} + \cdots + a^M z^{-M}.$$

Multiplying by $a\,z^{-1}$ gives
Equation:

$$a\,z^{-1}F\left(z\right) \;=\; a\,z^{-1} + a^{2}z^{-2} + a^{3}z^{-3} + a^{4}z^{-4} + \cdots + a^{M+1}z^{-M-1}$$

and subtracting from [link] gives
Equation:

$$\left(1 - a\,z^{-1}\right)F\left(z\right) \;=\; 1 - a^{M+1}z^{-M-1}$$

Solving for $F(z)$ results in
Equation:

$$F\left(z\right) \;=\; \frac{1 - a^{M+1}z^{-M-1}}{1 - a\,z^{-1}} \;=\; \frac{z - a\left(\frac{a}{z}\right)^{M}}{z - a}$$

The limit of this sum as $M \to \infty$ is
Equation:

$$F\left(z\right) \;=\; \frac{z}{z - a}$$

for $|z| > |a|$. This not only establishes the z-transform of $f(n)$ but gives the region in the z plane where the sum converges.

If a similar set of operations is performed on the sequence that exists for negative n
Equation:

$$f\left(n\right) \;=\; u\left(-n-1\right)a^{n} \;=\; \begin{cases} a^{n} & n < 0 \\ 0 & n \geq 0 \end{cases}$$

the result is
Equation:

$$F\left(z\right) \;=\; -\frac{z}{z - a}$$

for $|z| < |a|$. Here we have exactly the same z-transform for a different sequence $f(n)$ but with a different ROC. The pole in $F(z)$ divides the z-plane into two regions that give two different $f(n)$. This is a general result that can be applied to a general rational $F(z)$ with several poles and zeros. The z-plane will be divided into concentric annular regions separated by the poles. The contour integral is evaluated in one of these regions and the poles inside the contour give the part of the solution existing for negative n with the poles outside the contour giving the part of the solution existing for positive n.

Notice that any finite length signal has a z-transform that converges for all z. The ROC is the entire z-plane except perhaps zero and/or infinity.

Relation of the Z-Transform to the DTFT and the DFT

The FS coefficients are weights on the delta functions in a FT of the periodically extended signal. The FT is the LT evaluated on the imaginary axis: $s = j\omega$.

The DFT values are samples of the DTFT of a finite length signal. The DTFT is the z-transform evaluated on the unit circle in the z plane.
Equation:

$$F(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} = \mathscr{ZT}\{x(n)\}$$

Equation:

$$F\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n} = \mathscr{DTFT}\{x(n)\}$$

and if $x(n)$ is of length N
Equation:

$$F\left(e^{j\frac{2\pi}{N}k}\right) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi}{N}kn} = \mathscr{DFT}\{x(n)\}$$

It is important to be able to relate the time-domain signal $x(n)$, its spectrum $X(\omega)$, and its z-transform represented by the pole-zero locations on the z plane.

Relationships Among Fourier Transforms

The DFT takes a periodic discrete-time signal into a periodic discrete-frequency representation.

The DTFT takes a discrete-time signal into a periodic continuous-frequency representation.

The FS takes a periodic continuous-time signal into a discrete-frequency representation.

The FT takes a continuous-time signal into a continuous-frequency representation.

The LT takes a continuous-time signal into a function of a continuous complex variable.

The ZT takes a discrete-time signal into a function of a continuous complex variable.

Wavelet-Based Signal Analysis

There are wavelet systems and transforms analogous to the DFT, Fourier series, discrete-time Fourier transform, and the Fourier integral. We will start with the discrete wavelet transform (DWT) which is analogous to the Fourier series and probably should be called the wavelet series [link]. Wavelet analysis can be a form of time-frequency analysis which locates energy or events in time and frequency (or scale) simultaneously. It is somewhat similar to what is called a short-time Fourier transform or a Gabor transform or a windowed Fourier transform.

The history of wavelets and wavelet based signal processing is fairly recent. Its roots in signal expansion go back to early geophysical and image processing methods and in DSP to filter bank theory and subband coding. The current high interest probably started in the late 1980's with the work of Mallat, Daubechies, and others. Since then, the amount of research, publication, and application has exploded. Two excellent descriptions of the history of wavelet research and development are by Hubbard [link] and by Daubechies [link] and a projection into the future by Sweldens [link] and Burrus [link].

The Basic Wavelet Theory

The ideas and foundations of the basic dyadic, multiresolution wavelet systems are now pretty well developed, understood, and available [link], [link], [link], [link]. The **first** basic requirement is that a set of expansion functions (usually a basis) are generated from a single “mother” function by translation and scaling. For the discrete wavelet expansion system, this is

Equation:

$$\varphi_{j,k}(t) = \varphi\left(2^j t - k\right)$$

where j, k are integer indices for the series expansion of the form
Equation:

$$f(t) = \sum_{j,k} c_{j,k}\, \varphi_{j,k}(t).$$

The coefficients $c_{j,k}$ are called the discrete wavelet transform of the signal $f(t)$. This use of translation and scale to create an expansion system is the foundation of all so-called first generation wavelets [link].

The system is somewhat similar to the Fourier series described in Equation 51 from Least Squared Error Designed of FIR Filters with frequencies being related by powers of two rather than an integer multiple and the translation by k giving only the two results of cosine and sine for the Fourier series.

The **second** almost universal requirement is that the wavelet system generates a multiresolution analysis (MRA). This means that a low resolution function (low scale j) can be expanded in terms of the same function at a higher resolution (higher j). This is stated by requiring that the generator of a MRA wavelet system, called a scaling function $\varphi(t)$, satisfies
Equation:

$$\varphi(t) = \sum_n h(n)\, \varphi(2t - n).$$

This equation, called the **refinement equation** or the **MRA equation** or **basic recursion equation**, is similar to a differential equation in that its solution is what defines the basic scaling function and wavelet [link], [link].

The current state of the art is that most of the necessary and sufficient conditions on the coefficients $h(n)$ are known for the existence, uniqueness, orthogonality, and other properties of $\varphi(t)$. Some of the theory parallels Fourier theory and some does not.

A **third** important feature of a MRA wavelet system is a discrete wavelet transform (DWT) can be calculated by a digital filter bank using what is now called Mallat's algorithm. Indeed, this connection with digital signal processing (DSP) has been a rich source of ideas and methods. With this filter bank, one can calculate the DWT of a

length-N digital signal with order N operations. This means the number of multiplications and additions grows only linearly with the length of the signal. This compares with $N \log (N)$ for an FFT and N^2 for most methods and worse than that for some others.

These basic ideas came from the work of Meyer, Daubechies, Mallat, and others but for a time looked like a solution looking for a problem. Then a second phase of research showed there are many problems to which the wavelet is an excellent solution. In particular, the results of Donoho, Johnstone, Coifman, Beylkin, and others opened another set of doors.

Generalization of the Basic Wavelet System

After (in some cases during) much of the development of the above basic ideas, a number of generalizations [link] were made. They are listed below:

1. A larger integer scale factor than $M = 2$ can be used to give a more general **M-band** refinement equation [link]
   Equation:

$$\varphi(t) = \sum_n h(n)\, \varphi(Mt - n)$$

   than the “dyadic" or octave based Equation 4 from Rational Function Approximation. This also gives more than two channels in the accompanying filter bank. It allows a uniform frequency resolution rather than the resulting logarithmic one for $M = 2$.
2. The wavelet system called a **wavelet packet** is generated by “iterating" the wavelet branches of the filter bank to give a finer resolution to the wavelet decomposition. This was suggested by Coifman and it too allows a mixture of uniform and logarithmic frequency resolution. It also allows a relatively simple adaptive system to be developed which has an automatically adjustable frequency resolution based on the properties of the signal.
3. The usual requirement of translation orthogonality of the scaling function and wavelets can be relaxed to give what is called a **biorthogonal system**[link]. If the expansion basis is not orthogonal, a dual basis can be created that will allow the usual expansion and coefficient calculations to be made. The main disadvantage is the loss of a Parseval's theorem which maintains energy partitioning. Nevertheless, the greater flexibility of the biorthogonal system allows superior performance in many compression and denoising applications.
4. The basic refinement Equation 4 from Rational Function Approximation gives the scaling function in terms of a compressed version of itself (self-similar). If we

allow two (or more) scaling functions, each being a weighted sum of a compress version of both, a more general set of basis functions results. This can be viewed as a vector of scaling functions with the coefficients being a matrix now. Once again, this generalization allows more flexibility in the characteristics of the individual scaling functions and their related multi-wavelets. These are called **multi-wavelet systems** and are still being developed.

5. One of the very few disadvantages of the discrete wavelet transform is the fact it is not shift invariant. In other words, if you shift a signal in time, its wavelet transform not only shifts, it changes character! For many applications in denoising and compression, this is not desirable although it may be tolerable. The DWT can be made **shift-invariant** by calculating the DWT of a signal for all possible shifts and adding (or averaging) the results. That turns out to be equivalent to removing all of the down-samplers in the associated filter bank (an **undecimated filter bank**), which is also equivalent to building an overdetermined or **redundant DWT** from a traditional wavelet basis. This overcomplete system is similar to a “tight frame" and maintains most of the features of an orthogonal basis yet is shift invariant. It does, however, require $N \log (N)$ operations.
6. Wavelet systems are easily modified to being an adaptive system where the basis adjusts itself to the properties of the signal or the signal class. This is often done by starting with a large collection or library of expansion systems and bases. A subset is adaptively selected based on the efficiency of the representation using a process sometimes called **pursuit**. In other words, a set is chosen that will result in the smallest number of significant expansion coefficients. Clearly, this is signal dependent, which is both its strength and its limitation. It is nonlinear.
7. One of the most powerful structures yet suggested for using wavelets for signal processing is to first take the DWT, then do a point-wise linear or nonlinear processing of the DWT, finally followed by an inverse DWT. Simply setting some of the wavelet domain expansion terms to zero results in linear wavelet domain filtering, similar to what would happen if the same were done with Fourier transforms. Donoho [link], [link] and others have shown by using some form of nonlinear thresholding of the DWT, one can achieve near optimal denoising or compression of a signal. The concentrating or localizing character of the DWT allows this nonlinear thresholding to be very effective.

The present state of activity in wavelet research and application shows great promise based on the above generalizations and extensions of the basic theory and structure [link]. We now have conferences, workshops, articles, newsletters, books, and email groups that are moving the state of the art forward. More details, examples, and software are given in [link], [link], [link].

Discrete-Time Systems

In the context of discussing signal processing, the most general definition of a system is similar to that of a function. A system is a device, formula, rule, or some process that assigns an output signal from some given class to each possible input signal chosen from some allowed class. From this definition one can pose three interesting and practical problems.

1. **Analysis**: If the input signal and the system are given, find the output signal.
2. **Control**: If the system and the output signal are given, find the input signal.
3. **Synthesis**: If the input signal and output signal are given, find the system.

The definition of input and output signal can be quite diverse. They could be scalars, vectors, functions, functionals, or other objects.

All three of these problems are important, but analysis is probably the most basic and its study usually precedes that of the other two. Analysis usually results in a unique solution. Control is often unique but there are some problems where several inputs would give the same output. Synthesis is seldom unique. There are usually many possible systems that will give the same output for a given input.

In order to develop tools for analysis, control, and design of discrete-time systems, specific definitions, restrictions, and classifications must be made. It is the explicit statement of what a system is, not what it isn't, that allows a descriptive theory and design methods to be developed.

Classifications

The basic classifications of signal processing systems are defined and listed here. We will restrict ourselves to discrete-time systems that have ordered sequences of real or complex numbers as inputs and outputs and will denote the input sequence by $x(n)$ and the output sequence by $y(n)$ and show the process of the system by $x(n) \to y(n)$. Although the independent variable n could represent any physical variable, our most common usages causes us to generically call it time but the results obtained certainly are not restricted to this interpretation.

1. **Linear,** A system is classified as linear if two conditions are true.
    - If $x(n) \to y(n)$ then $a\,x(n) \to a\,y(n)$ for all a. This property is called homogeneity or scaling.

- If $x_1(n) \to y_1(n)$ and $x_2(n) \to y_2(n)$, then $(x_1(n) + x_2(n)) \to (y_1(n) + y_2(n))$ for all x_1 and x_2. This property is called superposition or additivity.

If a system does not satisfy both of these conditions for all inputs, it is classified as nonlinear. For most practical systems, one of these conditions implies the other. Note that a linear system must give a zero output for a zero input.

2. **Time Invariant** , also called index invariant or shift invariant. A system is classified as time invariant if $x(n+k) \to y(n+k)$ for any integer k. This states that the system responds the same way regardless of when the input is applied. In most cases, the system itself is not a function of time.
3. **Stable** . A system is called bounded-input bounded-output stable if for all bounded inputs, the corresponding outputs are bounded. This means that the output must remain bounded even for inputs artificially constructed to maximize a particular system's output.
4. **Causal** . A system is classified as causal if the output of a system does not precede the input. For linear systems this means that the impulse response of a system is zero for time before the input. This concept implies the interpretation of n as time even though it may not be. A system is semi-causal if after a finite shift in time, the impulse response is zero for negative time. If the impulse response is nonzero for $n \to -\infty$, the system is absolutely non-causal. Delays are simple to realize in discrete-time systems and semi-causal systems can often be made realizable if a time delay can be tolerated.
5. **Real-Time** . A discrete-time system can operate in “real-time" if an output value in the output sequence can be calculated by the system before the next input arrives. If this is not possible, the input and output must be stored in blocks and the system operates in “batch" mode. In batch mode, each output value can depend on all of the input values and the concept of causality does not apply.

These definitions will allow a powerful class of analysis and design methods to be developed and we start with convolution.

Convolution

The most basic and powerful operation for linear discrete-time system analysis, control, and design is discrete-time convolution. We first define the discrete-time unit impulse, also known as the Kronecker delta function, as

Equation:

$$\delta\left(n\right) = \begin{cases} 1 & \text{for } n = 0 \\ 0 & \text{otherwise.} \end{cases}$$

If a system is linear and time-invariant, and $\delta(n) \rightarrow h(n)$, the output $y(n)$ can be calculated from its input $x(n)$ by the operation called convolution denoted and defined by
Equation:

$$y\left(n\right) = h\left(n\right) {}^{*} x\left(n\right) = \sum_{m=-\infty}^{\infty} h\left(n-m\right) x\left(m\right)$$

It is informative to methodically develop this equation from the basic properties of a linear system.

Derivation of the Convolution Sum

We first define a complete set of orthogonal basis functions by $\delta(n-m)$ for $m = 0, 1, 2, \cdots, \infty$. The input $x(n)$ is broken down into a set of inputs by taking an inner product of the input with each of the basis functions. This produces a set of input components, each of which is a single impulse weighted by a single value of the input sequence $(x(n), \delta(n-m)) = x(m)\delta(n-m)$. Using the time invariant property of the system, $\delta(n-m) \rightarrow h(n-m)$ and using the scaling property of a linear system, this gives an output of $x(m)\delta(n-M) \rightarrow x(m)h(n-m)$. We now calculate the output due to $x(n)$ by adding outputs due to each of the resolved inputs using the superposition property of linear systems. This is illustrated by the following diagram:
Equation:

$$x\left(n\right) = \left\{ \begin{array}{lcll} x(n)\delta(n) & = & x(0)\delta(n) & \rightarrow \quad x(0)h(n) \\ x(n)\delta(n-1) & = & x(1)\delta(n-1) & \rightarrow \quad x(1)h(n-1) \\ x(n)\delta(n-2) & = & x(2)\delta(n-2) & \rightarrow \quad x(2)h(n-2) \\ \vdots & & & \quad \vdots \\ x(n)\delta(n-m) & = & x(m)\delta(n-m) & \rightarrow \quad x(m)h(n-m) \end{array} \right\} = y\left(n\right)$$

or
Equation:

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)\, h(n-m)$$

and changing variables gives
Equation:

$$y(n) = \sum_{m=-\infty}^{\infty} h(n-m)\, x(m)$$

If the system is linear but time varying, we denote the response to an impulse at $n = m$ by $\delta(n-m) \rightarrow h(n,m)$. In other words, each impulse response may be different depending on when the impulse is applied. From the development above, it is easy to see where the time-invariant property was used and to derive a convolution equation for a time-varying system as
Equation:

$$y(n) = h(n,m)^* x(n) = \sum_{m=-\infty}^{\infty} h(n,m) x(m).$$

Unfortunately, relaxing the linear constraint destroys the basic structure of the convolution sum and does not result in anything of this form that is useful.

By a change of variables, one can easily show that the convolution sum can also be written
Equation:

$$y(n) = h(n)^* x(n) = \sum_{m=-\infty}^{\infty} h(m) x(n-m).$$

If the system is causal, $h(n) = 0$ for $n < 0$ and the upper limit on the summation in Equation 2 from Discrete Time Signals becomes $m = n$. If the input signal is

causal, the lower limit on the summation becomes zero. The form of the convolution sum for a linear, time-invariant, causal discrete-time system with a causal input is
Equation:

$$y\left(n\right)=h\left(n\right)^{*}x\left(n\right)=\sum_{m=0}^{n}h\left(n-m\right)x\left(m\right)$$

or, showing the operations commute
Equation:

$$y\left(n\right)=h\left(n\right)^{*}x\left(n\right)=\sum_{m=0}^{n}h\left(m\right)x\left(n-m\right).$$

Convolution is used analytically to analyze linear systems and it can also be used to calculate the output of a system by only knowing its impulse response. This is a very powerful tool because it does not require any detailed knowledge of the system itself. It only uses one experimentally obtainable response. However, this summation cannot only be used to analyze or calculate the response of a given system, it can **be** an implementation of the system. This summation can be implemented in hardware or programmed on a computer and become the signal processor.

The Matrix Formulation of Convolution

Some of the properties and characteristics of convolution and of the systems it represents can be better described by a matrix formulation than by the summation notation. The first L values of the discrete-time convolution defined above can be written as a matrix operator on a vector of inputs to give a vector of the output values.
Equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \\ h_{L-1} & & \cdots & & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{L-1} \end{bmatrix}$$

If the input sequence x is of length N and the operator signal h is of length M, the output is of length $L = N + M - 1$. This is shown for $N = 4$ and $M = 3$ by the rectangular matrix operation

Equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0 \\ h_1 & h_0 & 0 & 0 \\ h_2 & h_1 & h_0 & 0 \\ 0 & h_2 & h_1 & h_0 \\ 0 & 0 & h_2 & h_1 \\ 0 & 0 & 0 & h_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

It is clear that if the system is causal ($h(n) = 0$ for $n < 0$), the H matrix is lower triangular. It is also easy to see that the system being time-invariant is equivalent to the matrix being Toeplitz [link]. This formulation makes it obvious that if a certain output were desired from a length 4 input, only 4 of the 6 values could be specified and the other 2 would be controlled by them.

Although the formulation of constructing the matrix from the impulse response of the system and having it operate on the input vector seems most natural, the matrix could have been formulated from the input and the vector would have been the impulse response. Indeed, this might the appropriate formulation if one were specifying the input and output and designing the system.

The basic convolution defined in [link], derived in [link], and given in matrix form in [link] relates the input to the output for linear systems. This is the form of convolution that is related to multiplication of the DTFT and z-transform of signals. However, it is cyclic convolution that is fundamentally related to the DFT and that will be efficiently calculated by the fast Fourier transform (FFT)

developed in Part III of these notes. Matrix formulation of length-L cyclic convolution is given by

Equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} h_0 & h_{L-1} & h_{L-2} & \cdots & h_1 \\ h_1 & h_0 & h_{L-1} & & h_2 \\ h_2 & h_1 & h_0 & & h_3 \\ \vdots & & & & \vdots \\ h_{L-1} & & \cdots & & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{L-1} \end{bmatrix}$$

This matrix description makes it clear that the matrix operator is always square and the three signals, $x(n)$, $h(n)$, and $y(n)$, are necessarily of the same length.

There are several useful conclusions that can be drawn from linear algebra [link]. The eigenvalues of the non-cyclic are all the same since the eigenvalues of a lower triangular matrix are simply the values on the diagonal.

Although it is less obvious, the eigenvalues of the cyclic convolution matrix are the N values of the DFT of $h(n)$ and the eigenvectors are the basis functions of the DFT which are the column vectors of the DFT matrix. The eigenvectors are completely controlled by the structure of H being a cyclic convolution matrix and are not at all a function of the values of $h(n)$. The DFT matrix equation from [link] is given by

Equation:

$$\mathbf{X} = \mathbf{F}\mathbf{x} \quad \text{and} \quad \mathbf{Y} = \mathbf{F}\mathbf{y}$$

where $\mathbf{X}$ is the length-N vector of the DFT values, $\mathbf{H}$ is the matrix operator for the DFT, and $\mathbf{x}$ is the length-N vector of the signal $x(n)$ values. The same is true for the comparable terms in y.

The matrix form of the length-N cyclic convolution in [link] is written

Equation:

$$\mathbf{y} = \mathbf{H}\mathbf{x}$$

Taking the DFT both sides and using the IDFT on x gives

Equation:

$$\mathbf{Fy} = \mathbf{Y} = \mathbf{FHx} = \mathbf{FHF^{-1}X}$$

If we define the diagonal matrix $\mathbf{H_d}$ as an L by L matrix with the values of the DFT of $h(n)$ on its diagonal, the convolution property of the DFT becomes
Equation:

$$\mathbf{Y} = \mathbf{H_dX}$$

This implies
Equation:

$$\mathbf{H_d} = \mathbf{FHF^{-1}} \quad \text{and} \quad \mathbf{H} = \mathbf{F^{-1}H_dF}$$

which is the basis of the earlier statement that the eigenvalues of the cyclic convolution matrix are the values of the DFT of $h(n)$ and the eigenvectors are the orthogonal columns of $\mathbf{F}$. The DFT matrix diagonalizes the cyclic convolution matrix. This is probably the most concise statement of the relation of the DFT to convolution and to linear systems.

An important practical question is how one calculates the non-cyclic convolution needed by system analysis using the cyclic convolution of the DFT. The answer is easy to see using the matrix description of H. The length of the output of non-cyclic convolution is $N + M - 1$. If $N - 1$ zeros are appended to the end of $h(n)$ and $M - 1$ zeros are appended to the end of $x(n)$, the cyclic convolution of these two augmented signals will produce exactly the same $N + M - 1$ values as non-cyclic convolution would. This is illustrated for the example considered before.
Equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & 0 & h_2 & h_1 \\ h_1 & h_0 & 0 & 0 & 0 & h_2 \\ h_2 & h_1 & h_0 & 0 & 0 & 0 \\ 0 & h_2 & h_1 & h_0 & 0 & 0 \\ 0 & 0 & h_2 & h_1 & h_0 & 0 \\ 0 & 0 & 0 & h_2 & h_1 & h_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ 0 \\ 0 \end{bmatrix}$$

Just enough zeros were appended so that the nonzero terms in the upper right-hand corner of $\mathbf{H}$ are multiplied by the zeros in the lower part of $\mathbf{x}$ and, therefore, do not contribute to $\mathbf{y}$. This does require convolving longer signals but the output is exactly what we want and we calculated it with the DFT-compatible cyclic convolution. Note that more zeros could have been appended to h and x and the first $N + M - 1$ terms of the output would have been the same only more calculations would have been necessary. This is sometimes done in order to use forms of the FFT that require that the length be a power of two.

If fewer zeros or none had been appended to h and x, the nonzero terms in the upper right-hand corner of $\mathbf{H}$, which are the “tail" of $h(n)$, would have added the values that would have been at the end of the non-cyclic output of $y(n)$ to the values at the beginning. This is a natural part of cyclic convolution but is destructive if non-cyclic convolution is desired and is called aliasing or folding for obvious reasons. Aliasing is a phenomenon that occurs in several arenas of DSP and the matrix formulation makes it easy to understand.

The Z-Transform Transfer Function

Although the time-domain convolution is the most basic relationship of the input to the output for linear systems, the z-transform is a close second in importance. It gives different insight and a different set of tools for analysis and design of linear time-invariant discrete-time systems.

If our system in linear and time-invariant, we have seen that its output is given by convolution.
Equation:

$$y\left(n\right) = \sum_{m=-\infty}^{\infty} h\left(n-m\right)x\left(m\right)$$

Assuming that $h(n)$ is such that the summation converges properly, we can calculate the output to an input that we already know has a special relation with discrete-time transforms. Let $x\left(n\right) = z^n$ which gives
Equation:

$$y(n) = \sum_{m=-\infty}^{\infty} h(n-m) z^m$$

With the change of variables of $k = n - m$, we have
Equation:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) z^{n-k} = \left[\sum_{k=-\infty}^{\infty} h(k) z^{-k} \right] z^n$$

or
Equation:

$$y(n) = H(z) z^n$$

We have the remarkable result that for an input of $x(n) = z^n$, we get an output of exactly the same form but multiplied by a constant that depends on z and this constant is the z-transform of the impulse response of the system. In other words, if the system is thought of as a matrix or operator, z^n is analogous to an eigenvector of the system and $H(z)$ is analogous to the corresponding eigenvalue.

We also know from the properties of the z-transform that convolution in the n domain corresponds to multiplication in the z domain. This means that the z-transforms of $x(n)$ and $y(n)$ are related by the simple equation
Equation:

$$Y(z) = H(z) X(z)$$

The z-transform decomposes $x(n)$ into its various components along z^n which passing through the system simply multiplies that value time $H(z)$ and the inverse z-transform recombines the components to give the output. This explains why the z-transform is such a powerful operation in linear discrete-time system theory. Its kernel is the eigenvector of these systems.

The z-transform of the impulse response of a system is called its transfer function (it transfers the input to the output) and multiplying it times the z-transform of the

input gives the z-transform of the output for any system and signal where there is a common region of convergence for the transforms.

Frequency Response of Discrete-Time Systems

The frequency response of a Discrete-Time system is something experimentally measurable and something that is a complete description of a linear, time-invariant system in the same way that the impulse response is. The frequency response of a linear, time-invariant system is defined as the magnitude and phase of the sinusoidal output of the system with a sinusoidal input. More precisely, if
Equation:

$$x(n) = \cos\left(\omega n\right)$$

and the output of the system is expressed as
Equation:

$$y(n) = M(\omega)\,\cos\left(\omega n + \varphi(\omega)\right) + T(n)$$

where $T(n)$ contains no components at ω, then $M(\omega)$ is called the magnitude frequency response and $\varphi(\omega)$ is called the phase frequency response. If the system is causal, linear, time-invariant, and stable, $T(n)$ will approach zero as $n \to \infty$ and the only output will be the pure sinusoid at the same frequency as the input. This is because a sinusoid is a special case of z^n and, therefore, an eigenvector.

If z is a complex variable of the special form
Equation:

$$z = e^{j\omega}$$

then using Euler's relation of $e^{jx} = \cos\left(x\right) + j\,\sin\left(x\right)$, one has
Equation:

$$x\left(n\right) = e^{j\omega n} = \cos\left(\omega n\right) + j\,\sin\left(\omega n\right)$$

and therefore, the sinusoidal input of (3.22) is simply the real part of z^n for a particular value of z, and, therefore, the output being sinusoidal is no surprise.

Fundamental Theorem of Linear, Time-Invariant Systems

The fundamental theorem of calculus states that an integral defined as an inverse derivative and one defined as an area under a curve are the same. The fundamental theorem of algebra states that a polynomial given as a sum of weighted powers of the independent variable and as a product of first factors of the zeros are the same. The fundamental theorem of arithmetic states that an integer expressed as a sum of weighted units, tens, hundreds, etc. or as the product of its prime factors is the same.

These fundamental theorems all state equivalences of different ways of expressing or calculating something. The fundamental theorem of linear, time-invariant systems states calculating the output of a system can be done with the impulse response by convolution or with the frequency response (or z-transform) with transforms. Stated another way, it says the frequency response can be found from directly calculating the output from a sinusoidal input or by evaluating the z-transform on the unit circle.
Equation:

$$\mathscr{Z}\{h(n)\}|_{z=e^{j\omega}} \quad = \quad A(\omega)\ e^{j\Theta(\omega)}$$

Pole-Zero Plots

Relation of PZ Plots, FR Plots, Impulse R

State Variable Formulation

Difference Equations

Flow Graph Representation

Standard Structures

FIR and IIR Structures

Quantization Effects

Multidimensional Systems

Sampling, Up--Sampling, Down--Sampling, and Multi--Rate

A very important and fundamental operation in discrete-time signal processing is that of sampling. Discrete-time signals are often obtained from continuous-time signal by simple sampling. This is mathematically modeled as the evaluation of a function of a real variable at discrete values of time [link]. Physically, it is a more complicated and varied process which might be modeled as convolution of the sampled signal by a narrow pulse or an inner product with a basis function or, perhaps, by some nonlinear process.

The sampling of continuous-time signals is reviewed in the recent books by Marks [link] which is a bit casual with mathematical details, but gives a good overview and list of references. He gives a more advanced treatment in [link]. Some of these references are [link], [link], [link], [link], [link], [link], [link]. These will discuss the usual sampling theorem but also interpretations and extensions such as sampling the value and one derivative at each point, or of non uniform sampling.

Multirate discrete-time systems use sampling and sub sampling for a variety of reasons [link], [link]. A very general definition of sampling might be any mapping of a signal into a sequence of numbers. It might be the process of calculating coefficients of an expansion using inner products. A powerful tool is the use of periodically time varying theory, particularly the bifrequency map, block formulation, commutators, filter banks, and multidimensional formulations. One current interest follows from the study of wavelet basis functions. What kind of sampling theory can be developed for signals described in terms of wavelets? Some of the literature can be found in [link], [link], [link], [link], [link].

Another relatively new framework is the idea of tight frames [link], [link], [link]. Here signals are expanded in terms of an over determined set of expansion functions or vectors. If these expansions are what is called a tight frame, the mathematics of calculating the expansion coefficients with inner products works just as if the expansion functions were an orthonormal basis set. The redundancy of tight frames offers interesting possibilities. One example of a tight frame is an over sampled band limited function expansion.

Fourier Techniques

We first start with the most basic sampling ideas based on various forms of Fourier transforms [link], [link], [link].

The Spectrum of a Continuous-Time Signal and the Fourier Transform

Although in many cases digital signal processing views the signal as simple sequence of numbers, here we are going to pose the problem as originating with a function of continuous time. The fundamental tool is the classical Fourier transform defined by
Equation:

$$F(\omega) = \int f(t)\, e^{-j\omega t}\, dt$$

and its inverse
Equation:

$$f(t) = \frac{1}{2\pi} \int F(\omega)\, e^{j\omega t}\, d\omega.$$

where $j = \sqrt{-1}$. The Fourier transform of a signal is called its spectrum and it is complex valued with a magnitude and phase.

If the signal is periodic with period $f(t) = f(t + P)$, the Fourier transform does not exist as a function (it may as a distribution) therefore the spectrum is defined as the set of Fourier series coefficients
Equation:

$$C(k) = \frac{1}{P} \int_0^P f(t)\, e^{-j2\pi kt/P}\, dt$$

with the expansion having the form
Equation:

$$f(t) = \sum_k C(k)\, e^{j2\pi kt/P}.$$

The functions $g_k(t) = e^{j2\pi kt/P}$ form an orthogonal basis for periodic functions and [link] is the inner product $C(k) = \langle f(t), g_k(t) \rangle$.

For the non-periodic case in [link] the spectrum is a function of continuous frequency and for the periodic case in [link], the spectrum is a number sequence (a function of discrete frequency).

The Spectrum of a Sampled Signal and the DTFT

The discrete-time Fourier transform (DTFT) as defined in terms samples of a continuous function is
Equation:

$$F_d(\omega) = \sum_n f(Tn) e^{-j\omega Tn}$$

and its inverse
Equation:

$$f(Tn) = \frac{T}{2\pi} \int_{-\pi/T}^{\pi/T} F_d(\omega) e^{j\omega Tn} d\omega$$

can be derived by noting that $F_d(\omega)$ is periodic with period $P = 2\pi/T$ and, therefore, it can be expanded in a Fourier series with [link] resulting from calculating the series coefficients using [link].

The spectrum of a discrete-time signal is defined as the DTFT of the samples of a continuous-time signal given in [link]. Samples of the signal are given by the inverse DTFT in [link] but they can also be obtained by directly sampling $f(t)$ in [link] giving
Equation:

$$f(Tn) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\omega) e^{j\omega Tn} d\omega$$

which can be rewritten as an infinite sum of finite integrals in the form
Equation:

$$f(Tn) = \frac{1}{2\pi} \sum_{\ell} \int_0^{2\pi/T} F(\omega + 2\pi\ell/T)\, e^{j(\omega+2\pi\ell/T)Tn}\, d\omega$$

Equation:

$$= \frac{1}{2\pi} \int_0^{2\pi/T} \left[\sum_{\ell} F(\omega + 2\pi\ell/T) \right] e^{j(\omega+2\pi\ell/T)Tn}\, d\omega$$

where $F_p(\omega)$ is a periodic function made up of shifted versions of $F(\omega)$ (aliased) defined in [link] Because [link] and [link] are equal for all Tn and because the limits can be shifted by π/T without changing the equality, the integrands are equal and we have
Equation:

$$F_d(\omega) = \frac{1}{T} \sum_{\ell} F(\omega + 2\pi\ell/T) = \frac{1}{T} F_p(\omega).$$

where $F_p(\omega)$ is a periodic function made up of shifted versions of $F(\omega)$ as in [link]. The spectrum of the samples of $f(t)$ is an aliased version of the spectrum of $f(t)$ itself. The closer together the samples are taken, the further apart the centers of the aliased spectra are.

This result is very important in determining the frequency domain effects of sampling. It shows what the sampling rate should be and it is the basis for deriving the sampling theorem.

Samples of the Spectrum of a Sampled Signal and the DFT

Samples of the spectrum can be calculated from a finite number of samples of the original continuous-time signal using the DFT. If we let the length of the DFT be N and separation of the samples in the frequency domain be Δ and define the periodic functions

Equation:

$$F_p\left(\omega\right) = \sum_{\ell} F\left(\omega + N\Delta\ell\right)$$

and
Equation:

$$f_p\left(t\right) = \sum_{m} f\left(t + NTm\right)$$

then from [link] and [link] samples of the DTFT of $f(Tn)$ are
Equation:

$$F_p\left(\Delta k\right) = T\sum_{n} f\left(Tn\right) e^{-jT\Delta nk}$$

Equation:

$$= T\sum_{m}\sum_{n=0}^{N-1} f\left(Tn + TNm\right) e^{-j\Delta(Tn+TNm)k}$$

Equation:

$$= T\sum_{n=0}^{N-1}\left[\sum_{m} f\left(Tn + TNm\right)\right] e^{-j\Delta(Tn+TNm)k},$$

therefore,
Equation:

$$F_p\left(\Delta k\right) \;=\; \mathscr{D}\mathscr{F}\mathscr{T}\left\{f_p\left(Tn\right)\right\}$$

if $\Delta TN = 2\pi$. This formula gives a method for approximately calculating values of the Fourier transform of a function by taking the DFT (usually with the FFT) of samples of the function. This formula can easily be verified by forming the Riemann sum to approximate the integrals in [link] or [link].

Samples of the DTFT of a Sequence

If the signal is discrete in origin and is not a sampled function of a continuous variable, the DTFT is defined with $T = 1$ as
Equation:

$$H\left(\omega\right) = \sum_{n} h\left(n\right) e^{-j\omega n}$$

with an inverse
Equation:

$$h\left(n\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H\left(\omega\right) e^{j\omega n} \, d\omega.$$

If we want to calculate $H(\omega)$, we must sample it and that is written as
Equation:

$$H\left(\Delta k\right) = \sum_{n} h\left(n\right) e^{-j\Delta k n}$$

which after breaking the sum into an infinite sum of length-N sums as was done in [link] becomes
Equation:

$$H\left(\Delta k\right) = \sum_{m} \sum_{n=0}^{N-1} h\left(n + Nm\right) e^{-j\Delta k n}$$

if $\Delta = 2\pi / N$. This allows us to calculate samples of the DTFT by taking the DFT of samples of a periodized $h(n)$.
Equation:

$$H\left(\Delta k\right) = \mathscr{DFT}\{h_p\left(n\right)\}.$$

This a combination of the results in [link] and in [link].

Fourier Series Coefficients from the DFT

If the signal to be analyzed is periodic, the Fourier integral in [link] does not converge to a function (it may to a distribution). This function is usually expanded in a Fourier series to define its spectrum or a frequency description. We will sample this function and show how to approximately calculate the Fourier series coefficients using the DFT of the samples.

Consider a periodic signal $\widetilde{f}(t) = \widetilde{f}(t+P)$ with N samples taken every T seconds to give $\widetilde{Tn}(t)$ for integer n such that $NT = P$. The Fourier series expansion of $\widetilde{f}(t)$ is
Equation:

$$\widetilde{f}(t) = \sum_{k=-\infty}^{\infty} C(k)\, e^{2\pi kt/P}$$

with the coefficients given in [link]. Samples of this are
Equation:

$$\widetilde{f}(Tn) = \sum_{k=-\infty}^{\infty} C(k)\, e^{2\pi kTn/P} = \sum_{k=-\infty}^{\infty} C(k)\, e^{2\pi kn/N}$$

which is broken into a sum of sums as
Equation:

$$\widetilde{f}(Tn) = \sum_{\ell-\infty}^{\infty} \sum_{k=0}^{N-1} C(k+N\ell)\, e^{2\pi(k+N\ell)n/N} = \sum_{k=0}^{N-1} \left[\sum_{\ell-\infty}^{\infty} C(k+N\ell) \right] e^{2\pi kn/N}.$$

But the inverse DFT is of the form
Equation:

$$\widetilde{f}(Tn) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)\, e^{j2\pi nk/N}$$

therefore,
Equation:

$$\mathscr{DFT}\left\{\widetilde{f}\left(Tn\right)\right\} = N\sum_{\ell} C\left(k + N\ell\right) = N\,C_p\left(k\right).$$

and we have our result of the relation of the Fourier coefficients to the DFT of a sampled periodic signal. Once again aliasing is a result of sampling.

Shannon's Sampling Theorem

Given a signal modeled as a real (sometimes complex) valued function of a real variable (usually time here), we define a bandlimited function as any function whose Fourier transform or spectrum is zero outside of some finite domain
Equation:

$$|F(\omega)| = 0 \;\text{ for }\; |\omega| > W$$

for some $W < \infty$. The sampling theorem states that if $f(t)$ is sampled
Equation:

$$f_s\left(n\right) = f\left(Tn\right)$$

such that $T < 2\pi/W$, then $f(t)$ can be exactly reconstructed (interpolated) from its samples $f_s\left(n\right)$ using
Equation:

$$f\left(t\right) = \sum_{n=-\infty}^{\infty} f_s\left(n\right)\left[\frac{\sin\left(\pi t/T - \pi n\right)}{\pi t/T - \pi n}\right].$$

This is more compactly written by defining the **sinc** function as
Equation:

$$\operatorname{sinc}\left(x\right) = \frac{\sin\left(x\right)}{x}$$

which gives the sampling formula Equation 53 from Least Squared Error Design of FIR Filters the form
Equation:

$$f(t) = \sum_n f_s(n) \operatorname{sinc}(\pi t/T - \pi n).$$

The derivation of Equation 53 from Least Squared Error Design of FIR Filters or Equation 56 from Least Squared Error Design of FIR Filters can be done a number of ways. One of the quickest uses infinite sequences of delta functions and will be developed later in these notes. We will use a more direct method now to better see the assumptions and restrictions.

We first note that if $f(t)$ is bandlimited and if $T < 2\pi/W$ then there is no overlap or aliasing in $F_p(\omega)$. In other words, we can write [link] as
Equation:

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega) e^{j\omega t}\, d\omega = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} F_p(\omega) e^{j\omega t}\, d\omega$$

but
Equation:

$$F_p(\omega) = \sum_\ell F(\omega + 2\pi\ell/T) = T\sum_n f(Tn) e^{-j\omega Tn}$$

therefore,
Equation:

$$f(t) = \frac{1}{2\pi}\int_{-\pi/T}^{\pi/T} \left[T\sum_n f(Tn) e^{-j\omega Tn}\right] e^{j\omega t}\, d\omega$$

Equation:

$$= \frac{T}{2\pi} \sum_n f(Tn) \int_{-\pi/T}^{\pi/T} e^{j(t-Tn)\omega} \, d\omega$$

Equation:

$$= \sum_n f(Tn) \, \frac{\sin\left(\frac{\pi}{T}t - \pi n\right)}{\frac{\pi}{T}t - \pi n}$$

which is the sampling theorem. An alternate derivation uses a rectangle function and its Fourier transform, the sinc function, together with convolution and multiplication. A still shorter derivation uses strings of delta function with convolutions and multiplications. This is discussed later in these notes.

There are several things to notice about this very important result. First, note that although $f(t)$ is defined for all t from only its samples, it does require an infinite number of them to exactly calculate $f(t)$. Also note that this sum can be thought of as an expansion of $f(t)$ in terms of an orthogonal set of basis function which are the sinc functions. One can show that the coefficients in this expansion of $f(t)$ calculated by an inner product are simply samples of $f(t)$. In other words, the sinc functions span the space of bandlimited functions with a very simple calculation of the expansion coefficients. One can ask the question of what happens if a signal is “under sampled". What happens if the reconstruction formula in Equation 12 from Continuous Time Signals is used when there is aliasing and Equation 57 from Least Squarred Error Design of FIR Filters is not true. We will not pursue that just now. In any case, there are many variations and generalizations of this result that are quite interesting and useful.

Calculation of the Fourier Transform and Fourier Series using the FFT

Most theoretical and mathematical analysis of signals and systems use the Fourier series, Fourier transform, Laplace transform, discrete-time Fourier transform (DTFT), or the z-transform, however, when we want to actually evaluate transforms, we calculate values at sample frequencies. In other words, we use the discrete Fourier transform (DFT) and, for efficiency, usually evaluate it with the FFT algorithm. An important question is how can we calculate or approximately calculate these symbolic formula-based transforms with our

practical finite numerical tool. It would certainly seem that if we wanted the Fourier transform of a signal or function, we could sample the function, take its DFT with the FFT, and have some approximation to samples of the desired Fourier transform. We saw in the previous section that it is, in fact, possible provided some care is taken.

Summary

For the signal that is a function of a continuous variable we have
Equation:

$$\begin{array}{lrcl} \text{FT:} & f(t) & \rightarrow & F(\omega) \\ \text{DTFT:} & f(Tn) & \rightarrow & \frac{1}{T}\,F_p\big(\omega\big) = \frac{1}{T}\sum_\ell F\big(\omega + 2\pi\ell/T\big) \\ \text{DFT:} & f_p(Tn) & \rightarrow & \frac{1}{T}\,F_p\big(\Delta k\big) \;\text{ for }\; \Delta T N = 2\pi \end{array}$$

For the signal that is a function of a discrete variable we have
Equation:

$$\begin{array}{lrcl} \text{DTFT:} & h(n) & \rightarrow & H(\omega) \\ \text{DFT:} & h_p(n) & \rightarrow & H(\Delta k) \text{for} \Delta N = 2\pi \end{array}$$

For the periodic signal of a continuous variable we have
Equation:

$$\begin{array}{lrcl} \text{FS:} & \tilde{g}(t) & \rightarrow & C(k) \\ \text{DFT:} & \tilde{g}(Tn) & \rightarrow & N\,C_p(k) \;\text{ for }\; TN = P \end{array}$$

For the sampled bandlimited signal we have
Equation:

$$\begin{array}{lrcl} \text{Sinc:} & f(t) & \rightarrow & f(Tn) \\ & & & f(t) = \sum_n f(Tn)\,\text{sinc}(2\pi t/T - \pi n) \\ & & & \text{if }\; F(\omega) = 0 \;\text{ for }\; |\omega| > 2\pi/T \end{array}$$

These formulas summarize much of the relations of the Fourier transforms of sampled signals and how they might be approximately calculate with the FFT. We next turn to the use of distributions and strings of delta functions as tool to study sampling.

Sampling Functions — the Shah Function

Th preceding discussions used traditional Fourier techniques to develop sampling tools. If distributions or delta functions are allowed, the Fourier transform will exist for a much larger class of signals. One should take care when using distributions as if they were functions but it is a very powerful extension.

There are several functions which have equally spaced sequences of impulses that can be used as tools in deriving a sampling formula. These are called “pitch fork" functions, picket fence functions, comb functions and shah functions. We start first with a finite length sequence to be used with the DFT. We define
Equation:

$$\text{Ш}_M(n) = \sum_{m=0}^{L-1} \delta(n - Mm)$$

where $N = LM$.
Equation:

$$DFT\{\text{Ш}_M(n)\} = \sum_{n=0}^{N-1} \left[\sum_{m=0}^{L-1} \delta(n - Mm) \right] e^{-j2\pi nk/N}$$

Equation:

$$= \sum_{m=0}^{L-1} \left[\sum_{n=0}^{N-1} \delta(n - Mm) e^{-j2\pi nk/N} \right]$$

Equation:

$$= \sum_{m=0}^{L-1} e^{-j2\pi Mmk/N} = \sum_{m=0}^{L-1} e^{-j2\pi mk/L}$$

Equation:

$$= \begin{cases} L & < k > L = 0 \\ 0 & \text{otherwise} \end{cases}$$

Equation:

$$= L \sum_{l=0}^{M-1} \delta \left(k - Ll \right) = L \amalg_L \left(k \right)$$

For the DTFT we have a similar derivation:
Equation:

$$DTFT \left\{ \amalg_M \left(n \right) \right\} = \sum_{n=-\infty}^{\infty} \left[\sum_{m=0}^{L-1} \delta \left(n - Mm \right) \right] e^{-j\omega n}$$

Equation:

$$= \sum_{m=0}^{L-1} \left[\sum_{n=-\infty}^{\infty} \delta \left(n - Mm \right) e^{-j\omega n} \right]$$

Equation:

$$= \sum_{m=0}^{L-1} e^{-j\omega Mm}$$

Equation:

$$= \begin{cases} L & \omega = k2\pi/M \\ 0 & \text{otherwise} \end{cases}$$

Equation:

$$= \sum_{l=0}^{M-1} \delta\left(\omega - 2\pi l/Ml\right) = K\, \text{Ш}_{2\pi/\text{M}}\left(\omega\right)$$

where K is constant.

An alternate derivation for the DTFT uses the inverse DTFT.
Equation:

$$IDTFT\left\{\text{Ш}_{2\pi/M}\left(\omega\right)\right\} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \text{Ш}_{2\pi/M}\left(\omega\right) e^{j\omega n}\, d\omega$$

Equation:

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} \sum_{l} \delta\left(\omega - 2\pi l/M\right) e^{j\omega n}\, d\omega$$

Equation:

$$= \frac{1}{2\pi}\sum_{l}\int_{-\pi}^{\pi} \delta\left(\omega - 2\pi l/M\right) e^{j\omega n}\, d\omega$$

Equation:

$$= \frac{1}{2\pi}\sum_{l=0}^{M-1} e^{2\pi l n/M} = \begin{cases} M/2\pi & n = M \\ 0 & \text{otherwise} \end{cases}$$

Equation:

$$= \left(\frac{M}{2\pi}\right)\text{Ш}_{2\pi/\text{M}}\left(\omega\right)$$

Therefore,
Equation:

$$\text{Ш}_{M}\left(n\right) \rightarrow \left(\frac{2\pi}{M}\right)\text{Ш}_{2\pi/\text{T}}\left(\omega\right)$$

For regular Fourier transform, we have a string of impulse functions in both the time and frequency. This we see from:
Equation:

$$FT\left\{\amalg_T\left(t\right)\right\} = \int_{-\infty}^{\infty} \sum_n \delta\left(t - nT\right) e^{-j\omega t}\, dt = \sum_n \int \delta\left(t - nT\right) e^{-j\omega t}\, dt$$

Equation:

$$= \sum_n e^{-j\omega nT} = \begin{cases} \infty & \omega = 2\pi/T \\ 0 & \text{otherwise} \end{cases}$$

Equation:

$$= \frac{2\pi}{T}\, \amalg_{2\pi/\mathrm{T}}\left(\omega\right)$$

The multiplicative constant is found from knowing the result for a single delta function.

These “shah functions" will be useful in sampling signals in both the continuous time and discrete time cases.

Up–Sampling, Signal Stretching, and Interpolation

In several situations we would like to increase the data rate of a signal or, to increase its length if it has finite length. This may be part of a multi rate system or part of an interpolation process. Consider the process of inserting $M - 1$ zeros between each sample of a discrete time signal.
Equation:

$$y\left(n\right) = \begin{cases} x(n/M) & < n > M = 0 (\text{or } \ n \ = \mathrm{kM}) \\ 0 & \text{otherwise} \end{cases}$$

For the finite length sequence case we calculate the DFT of the stretched or up–sampled sequence by
Equation:

$$C_s(k) = \sum_{n=0}^{MN-1} y(n) W_{MN}^{nk}$$

Equation:

$$C_s(k) = \sum_{n=0}^{MN-1} x(n/M)\ \text{Ш}_M(n) W_{MN}^{nk}$$

where the length is now NM and $k = 0, 1, \cdots, NM-1$. Changing the index variable $n = Mm$ gives:
Equation:

$$C_s(k) = \sum_{m=0}^{N-1} x(m) W_N^{mk} = C(k).$$

which says the DFT of the stretched sequence is exactly the same as the DFT of the original sequence but over M periods, each of length N.

For up–sampling an infinitely long sequence, we calculate the DTFT of the modified sequence in Equation 34 from FIR Digital Filters as
Equation:

$$C_s(\omega) = \sum_{n=-\infty}^{\infty} x(n/M)\ \text{Ш}_M(n) e^{-j\omega n} = \sum_m x(m) e^{-j\omega Mm}$$

Equation:

$$= C(M\omega)$$

where $C(\omega)$ is the DTFT of $x(n)$. Here again the transforms of the up–sampled signal is the same as the original signal except over M periods. This shows up here as $C_s(\omega)$ being a compressed version of M periods of $C(\omega)$.

The z-transform of an up–sampled sequence is simply derived by:
Equation:

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n) z^{-n} = \sum_{n} x(n/M) \amalg_M(n) z^{-n} = \sum_{m} x(m) z^{-Mm}$$

Equation:

$$= X\left(z^M\right)$$

which is consistent with a complex version of the DTFT in [link].

Notice that in all of these cases, there is no loss of information or invertibility. In other words, there is no aliasing.

Down–Sampling, Subsampling, or Decimation

In this section we consider the sampling problem where, unless there is sufficient redundancy, there will be a loss of information caused by removing data in the time domain and aliasing in the frequency domain.

The sampling process or the down sampling process creates a new shorter or compressed signal by keeping every M^{th} sample of the original sequence. This process is best seen as done in two steps. The first is to mask off the terms to be removed by setting $M-1$ terms to zero in each length-M block (multiply $x(n)$ by $\amalg_M(n)$), then that sequence is compressed or shortened by removing the $M-1$ zeroed terms.

We will now calculate the length $L = N/M$ DFT of a sequence that was obtained by sampling every M terms of an original length-N sequence $x(n)$. We will use the orthogonal properties of the basis vectors of the DFT which says:
Equation:

$$\sum_{n=0}^{M-1} e^{-j2\pi nl/M} = \begin{cases} M & \text{if n is an integer multiple of} M \\ 0 & \text{otherwise.} \end{cases}$$

We now calculate the DFT of the down-sampled signal.
Equation:

$$C_d\left(k\right) = \sum_{m=0}^{L-1} x\left(Mm\right) W_L^{mk}$$

where $N = LM$ and k = 0,1,...,L-1. This is done by masking $x(n)$.
Equation:

$$C_d\left(k\right) = \sum_{n=0}^{N-1} x\left(n\right) x_M\left(n\right) W_L^{nk}$$

Equation:

$$= \sum_{n=0}^{N-1} x\left(n\right) \left[\frac{1}{M} \sum_{l=0}^{M-1} e^{-j2\pi nl/M}\right] e^{-j2\pi nk/N}$$

Equation:

$$= \frac{1}{M} \sum_{l=0}^{M-1} \sum_{n=0}^{N-1} x\left(n\right) e^{j2\pi (k+Ll)n/N}$$

Equation:

$$= \frac{1}{M} \sum_{l=0}^{M-1} C\left(k+Ll\right)$$

The compression or removal of the masked terms is achieved in the frequency domain by using k = 0,1,...,L-1 This is a length-L DFT of the samples of $x(n)$. Unless $C(k)$ is sufficiently bandlimited, this causes aliasing and $x(n)$ is not unrecoverable.

It is instructive to consider an alternative derivation of the above result. In this case we use the IDFT given by
Equation:

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} C(k) W_N^{-nk}.$$

The sampled signal gives
Equation:

$$y(n) = x(Mn) = \frac{1}{N} \sum_{k=0}^{N-1} C(k) W_N^{-Mnk}.$$

for $n = 0, 1, \cdots, L-1$. This sum can be broken down by
Equation:

$$y(n) = \frac{1}{N} \sum_{k=0}^{L-1} \sum_{l=0}^{M-1} C(k+Ll) W_N^{-Mn(k+Ll)}.$$

Equation:

$$= \frac{1}{N} \sum_{k=0}^{L-1} \left[\sum_{l=0}^{M-1} C(k+Ll) \right] W_N^{-Mnk}.$$

From the term in the brackets, we have
Equation:

$$C_s(k) = \sum_{l=0}^{M-1} C(k+Ll)$$

as was obtained in [link].

Now consider still another derivation using shah functions. Let
Equation:

$$x_s(n) = \text{Ш}_M(n) x(n)$$

From the convolution property of the DFT we have
Equation:

$$C_s\left(k\right) = L\ \text{Ш}_L\left(k\right) * C\left(k\right)$$

therefore
Equation:

$$C_s\left(k\right) = \sum_{l=0}^{M-1} C\left(k + Ll\right)$$

which again is the same as in [link].

We now turn to the down sampling of an infinitely long signal which will require use of the DTFT of the signals.
Equation:

$$C_s\left(\omega\right) = \sum_{m=-\infty}^{\infty} x\left(Mm\right) e^{-j\omega Mm}$$

Equation:

$$= \sum_{n} x\left(n\right)\ \text{Ш}_M\left(n\right) e^{-j\omega n}$$

Equation:

$$= \sum_{n} x\left(n\right) \left[\frac{1}{M} \sum_{l=0}^{M-1} e^{-j2\pi nl/M}\right] e^{-j\omega n}$$

Equation:

$$= \frac{1}{M} \sum_{l=0}^{M-1} \sum_{n} x\left(n\right) e^{-j(\omega - 2\pi l/M)n}$$

Equation:

$$= \frac{1}{M} \sum_{l=0}^{M-1} C\left(\omega - 2\pi l/M\right)$$

which shows the aliasing caused by the masking (sampling without compression). We now give the effects of compressing $x_s\left(n\right)$ which is a simple scaling of ω. This is the inverse of the stretching results in [link].
Equation:

$$C_s\left(\omega\right) = \frac{1}{M} \sum_{l=0}^{M-1} C\left(\omega/M - 2\pi l/M\right).$$

In order to see how the various properties of the DFT can be used, consider an alternate derivation which uses the IDTFT.
Equation:

$$x\left(n\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} C\left(\omega\right) e^{j\omega n}\, d\omega$$

which for the down–sampled signal becomes
Equation:

$$x\left(Mn\right) = \frac{1}{2\pi} \int_{-\pi}^{\pi} C\left(\omega\right) e^{j\omega Mn}\, d\omega$$

The integral broken into the sum of M sections using a change of variables of $\omega = \left(\omega_1 + 2\pi l\right)/M$ giving
Equation:

$$x\left(Mn\right) = \frac{1}{2\pi} \sum_{l=0}^{M-1} \int_{-\pi}^{\pi} C\left(\omega_1/M + 2\pi l/M\right) e^{j(\omega_1/M + 2\pi l/M)Mn}\, d\omega_1$$

which shows the transform to be the same as given in Equation 9 from Chebyshev of Equal Ripple Error Approximation Filters.

Still another approach which uses the shah function can be given by
Equation:

$$x_s(n) = \text{Ш}_M(n)\, x(n)$$

which has as a DTFT
Equation:

$$C_s(\omega) = \left(\frac{2\pi}{M}\right) \text{Ш}_{2\pi/M}(\omega) * C(\omega)$$

Equation:

$$= \frac{2\pi}{M} \sum_{l=0}^{M-1} C(\omega + 2\pi l/M)$$

which after compressing becomes
Equation:

$$C_s = \frac{2\pi}{M} \sum_{l=0}^{M-1} C(\omega/M + 2\pi l/M)$$

which is same as Equation 9 from Chebyshev of Equal Ripple Error Approximation Filters.

Now we consider the effects of down–sampling on the z-transform of a signal.
Equation:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n}$$

Applying this to the sampled signal gives
Equation:

$$X_s(z) = \sum_n x(Mn)\, z^{-Mn} = \sum_n x(n)\ \text{Ш}_M(n)\, z^{-n}$$

Equation:

$$= \sum_{n} x\left(n\right) \sum_{l=0}^{M-1} e^{j2\pi nl/M} \, z^{-n}$$

Equation:

$$= \sum_{l=0}^{M-1} \sum_{n} x\left(n\right) \left\{ e^{j2\pi l/M} \, z \right\}^{-n}$$

Equation:

$$= \sum_{l=0}^{M-1} X\left(e^{-j2\pi l/M} \, z \right)$$

which becomes after compressing
Equation:

$$= \sum_{l=0}^{M-1} X\left(e^{-j2\pi l/M} \, z^{1/M} \right).$$

This concludes our investigations of the effects of down–sampling a discrete–time signal and we discover much the same aliasing properties as in sampling a continuous–time signal. We also saw some of the mathematical steps used in the development.

More Later

We will later develop relations of sampling to multirate systems, periodically time varying systems, and block processing. This should be a very effective formulation for teaching as well as research on these topics.

FIR Digital Filters

There are two types of linear, time-invariant digital filters. We will investigate digital filters with a **finite-duration impulse response** (FIR) in this section and those with an **infinite-duration impulse response** (IIR) in another document. FIR filters have characteristics that make them useful in many applications [link], [link].

1. FIR filters can achieve an exactly linear phase frequency response
2. FIR filters cannot be unstable.
3. FIR filters are generally less sensitive to coefficient round-off and finite-precision arithmetic than IIR filters.
4. FIR filters design methods are generally linear.
5. FIR filters can be efficiently realized on general or special-purpose hardware.

However, frequency responses that need a rapid transition between bands and do not require linear phase are often more efficiently realized with IIR filters.

It is the purpose of this section to examine and evaluate these characteristics which are important in the design of the four basic types of linear-phase FIR filters.

Because of the usual methods of implementation, the Finite Impulse Response (FIR) filter is also called a **nonrecursive** filter or a convolution filter. From the time-domain view of this operation, the FIR filter is sometimes called a **moving-average** or **running-average** filter. All of these names represent useful interpretations that are discussed in this section; however, the name, FIR, is most commonly seen in filter-design literature and is used in these notes.

The duration or sequence length of the impulse response of these filters is by definition finite; therefore, the output can be written as a finite convolution sum by
Equation:

$$y\left(n\right) \; = \; \sum_{m=0}^{N-1} h\left(m\right) x\left(n-m\right)$$

where n and m are integers, perhaps representing samples in time, and where $x(n)$ is the input sequence, $y(n)$ the output sequence, and $h(n)$ is the length-N impulse response of the filter. With a change of index variables, this can also be written as
Equation:

$$y\left(n\right) \; = \; \sum_{m=n}^{n-N+1} h\left(n-m\right) x\left(m\right).$$

If the FIR filter is interpreted as an extension of a moving sum or as a weighted moving average, some of its properties can easily be seen. If one has a sequence of numbers, e.g., prices from the daily stock market for a particular stock, and would like to remove the erratic variations in order to discover longer term trends, each number could be replaced by the average of itself and the preceding three numbers, i.e., the variations within a four-day period would be “averaged out" while the longer-term variations would remain. To illustrate how this happens, consider an artificial signal $x(n)$ containing a linear term, $K_1 n$, and an undesired oscillating term added to it, such that
Equation:

$$x\left(n\right) \; = \; K_1 n + K_2 \cos\left(\pi n\right)$$

If a length-2 averaging filter is used with
Equation:

$$h\left(n\right) \ = \ \begin{cases} 1/2 & \text{for } \ n = 0, 1 \\ 0 & \text{otherwise} \end{cases}$$

it can be verified that, after two outputs, the output $y(n)$ is exactly the linear term $x(n)$ with a delay of one half sample interval and no oscillation.

This example illustrates the basic FIR filter-design problem: determine N, the number of terms for $h(n)$, and the values of $h(n)$ for achieving a desired effect on the signal. The reader should examine simple examples to obtain an intuitive idea of the FIR filter as a moving average; however, this simple time-domain interpretation will not suffice for complicated problems where the concept of frequency becomes more valuable.

Frequency-Domain Description of FIR Filters

The output of a length-N FIR filter can be calculated from the input using convolution.
Equation:

$$y\left(n\right) = \sum_{k=0}^{N-1} h\left(k\right) \ x\left(n-k\right)$$

and the transfer function of an FIR filter is given by the z-transform of the finite length impulse response $h(n)$ as
Equation:

$$H\left(z\right) \ = \ \sum_{n=0}^{N-1} h\left(n\right) z^{-n}.$$

The frequency response of a filter, is found by setting $z = e^{j\omega}$, which is the same as the discrete-time Fourier transform (DTFT) of $h(n)$, which gives
Equation:

$$H\left(\omega\right) \ = \ \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega n}$$

with ω being frequency in radians per second. Strictly speaking, the exponent should be $-j\omega Tn$ where T is the time interval between the integer steps of n (the sampling interval). But to simplify notation, it will be assumed that $T = 1$ until later in the notes where the relation between n and time is more important. Also to simplify notation, $H(\omega)$ is used to represent the frequency response rather that $H\left(e^{j\omega}\right)$. It should always be clear from the context whether H is a function of z or ω.

This frequency-response function is complex-valued and consists of a magnitude and a phase. Even though the impulse response is a function of the discrete variable n, the frequency response is a function of the continuous-frequency variable ω and is periodic with period 2π. This periodicity is easily shown by
Equation:

$$\begin{aligned} H(w+2\pi) &= \sum_{n=0}^{N-1} h\left(n\right) e^{-j(w+2\pi)n} \\ &= \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega n} e^{-j2\pi n} \; = \; H\left(\omega\right) \end{aligned}$$

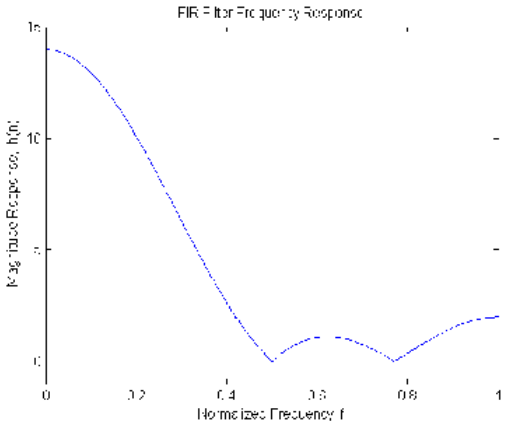
with frequency denoted by ω in radians per second or by f in Hz (hertz or cycles per second). These are related by
Equation:

$$\omega \; = \; 2\pi f$$

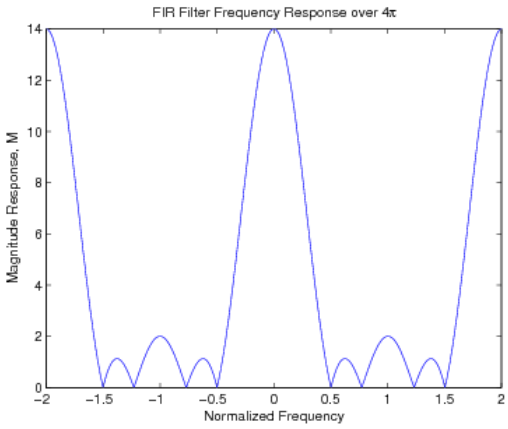
An example of a length-5 filter might be
Equation:

$$h(n) \; = \; 2, 3, 4, 3, 2$$

with a frequency-response plot shown over the base frequency band ($0 < \omega < \pi$ or $0 < f < 1$ in [link]. To illustrate the periodic nature of the total frequency response, [link] shows the response over a wider set of frequencies.



Frequency Response of Example Filter



Frequency Response of Example Filter over a wide band of frequencies

The Discrete Fourier Transform (DFT) can be used to evaluate the frequency response at certain frequencies. The DFT [link] of the length-N impulse response $h(n)$ is defined as
Equation:

$$C(k) = \sum_{n=0}^{N-1} h(n) e^{-j2\pi nk/N} \qquad k = 0, 1, \ldots, N-1$$

which, when compared to [link], gives
Equation:

$$C(k) = H(\omega_k) = H(2\pi k/N) \qquad k = 0, 1, \ldots, N-1$$

for $\omega_k = 2\pi k/N$.

This states that the DFT of $h(n)$ gives N samples of the frequency-response function $H(\omega)$. This sampling at N points may not give enough detail, and, therefore, more samples are needed. Any number of equally spaced samples can be found with the DFT by simply appending $L - N$ zeros to $h(n)$ and taking an L-length DFT. This is often useful when an accurate picture of all of $H(\omega)$ is required. Indeed, when the number of appended zeros goes to infinity, the DFT becomes the discrete-time Fourier transform of $h(n)$.

The fact that the DFT of $h(n)$ is a set of N samples of the frequency response suggests a method of designing FIR filters in which the inverse DFT of N samples of a desired frequency response gives the filter coefficients $h(n)$. That approach is called frequency sampling and is developed in another section.

Linear-Phase FIR Filters

A particular property of FIR filters that has proven to be very powerful is that a linear phase shift for the frequency response is possible. This is especially important to time domain details of a signal. The spectrum of a signal contains the individual frequency domain components separated in frequency. The process of filtering usually involves passing some of these components and rejecting others. This is done by multiplying the desired ones by one and the undesired ones by zero. When they are recombined, it is important that the

components have the same time domain alignment as they originally did. That is exactly what linear phase insures. A phase response that is linear with frequency keeps all of the frequency components properly registered with each other. That is especially important in seismic, radar, and sonar signal analysis as well as for many medical signals where the relative time locations of events contains the information of interest.

To develop the theory for linear phase FIR filters, a careful definition of phase shift is necessary. If the real and imaginary parts of $H(\omega)$ are given by
Equation:

$$H(\omega) \; = \; R(\omega) + jI(\omega)$$

where $j = \sqrt{-1}$ and the magnitude is defined by
Equation:

$$|H\left(\omega\right)| \; = \; \sqrt{R^2 + I^2}$$

and the phase by
Equation:

$$\Phi(\omega) \; = \arctan\left(I/R\right)$$

which gives
Equation:

$$H\left(\omega\right) \; = \; |H\left(\omega\right)|e^{j\Phi(\omega)}$$

in terms of the magnitude and phase. Using the real and imaginary parts is using a rectangular coordinate system and using the magnitude and phase is using a polar coordinate system. Often, the polar system is easier to interpret.

Mathematical problems arise from using $|H(\omega)|$ and $\Phi(\omega)$, because $|H(\omega)|$ is not analytic and $\Phi(\omega)$ not continuous. This problem is solved by introducing an amplitude function $A(\omega)$ that is real valued and may be positive or negative. The frequency response is written as
Equation:

$$H\left(\omega\right) \; = \; A\left(\omega\right)e^{j\Theta(\omega)}$$

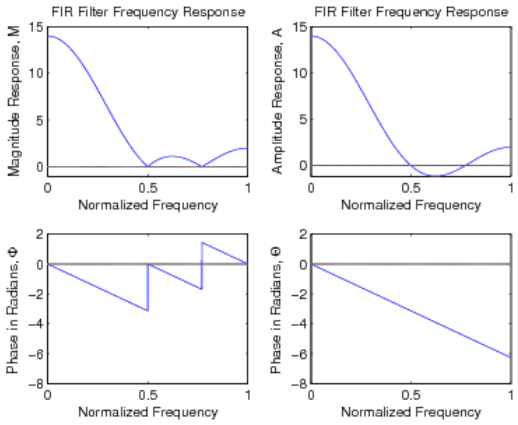
where $A(\omega)$ is called the amplitude in order to distinguish it from the magnitude $|H(\omega)|$, and $\Theta(\omega)$ is the continuous version of $\Phi(\omega)$. $A(\omega)$ is a real, analytic function that is related to the magnitude by
Equation:

$$A(\omega) \; = \; \pm|H(\omega)|$$

or
Equation:

$$|A(\omega)| \; = \; |H(\omega)|$$

With this definition, $A(\omega)$ can be made analytic and $\Theta(\omega)$ continuous. These are much easier to work with than $|H(\omega)|$ and $\Phi(\omega)$. The relationship of $A(\omega)$ and $|H(\omega)|$, and of $\Theta(\omega)$ and $\Phi(\omega)$ are shown in [link].



Magnitude and Amplitude Frequency Responses and Corresponding Phase Frequency Response of Example Filter

To develop the characteristics and properties of linear-phase filters, assume a general linear plus constant form for the phase function as
Equation:

$$\Theta(\omega) = K_1 + K_2\omega$$

This gives the frequency response function of a length-N FIR filter as
Equation:

$$\begin{aligned} H(\omega) &= \sum_{n=0}^{N-1} h(n)e^{-j\omega n} \\ &= e^{-j\omega M}\sum_{n=0}^{N-1} h(n)e^{j\omega(M-n)} \end{aligned}$$

and
Equation:

$$H(\omega) = e^{-j\omega M}\left[h_0 e^{j\omega M} + h_1 e^{j\omega(M-1)} + \cdots + h_{N-1}e^{j\omega(M-N+1)}\right]$$

[link] can be put in the form of
Equation:

$$H(\omega) = A(\omega)e^{j(K_1+K_2\omega)}$$

if M (not necessarily an integer) is defined by
Equation:

$$M \;=\; \frac{N-1}{2}$$

or equivalently,
Equation:

$$M \;=\; N-M-1$$

[link] then becomes
Equation:

$$\begin{aligned} H(\omega) \;\;=\;\; & e^{-j\omega M}\big[(h_0+h_{N-1})\cos{(\omega M)}+j\,(h_0-h_{N-1})\sin{(\omega M)} \\ & +(h_1+h_{N-2})\cos{(\omega\,(M-1))}+j\,(h_1-h_{N-2})\sin{(w\,(M-1))}+\;\cdots\;\big] \end{aligned}$$

There are two possibilities for putting this in the form of [link] where $A(\omega)$ is real: $K_1=0$ or $K_1=\pi/2$. The first case requires a special even symmetry in $h(n)$ of the form
Equation:

$$h(n) \;=\; h(N-n-1)$$

which gives
Equation:

$$H\,(\omega) \;=\; A\,(\omega)e^{-jM\omega}$$

where $A(\omega)$ is the amplitude, a real-valued function of ω and $e^{-jM\omega}$ gives the linear phase with M being the group delay. For the case where N is odd, using [link], [link], and [link], we have
Equation:

$$A\,(\omega) \;=\; \sum_{n=0}^{M-1} 2h\,(n)\cos{\omega\,(M-n)}+h\,(M)$$

or with a change of variables,
Equation:

$$A\,(\omega) \;=\; \sum_{n=1}^{M} 2h\,(M-n)\cos{(\omega n)}+h\,(M)$$

which becomes
Equation:

$$A\,(\omega) \;=\; \sum_{n=1}^{M} 2\widehat{h}\,(n)\cos{(\omega n)}+h\,(M)$$

where $\widehat{h}\,(n) = h\,(M-n)$ is a shifted $h(n)$. These formulas can be made simpler by defining new coefficients so that [link] becomes
Equation:

$$A\,(\omega) \;=\; \sum_{n=0}^{M} a\,(n)\;\cos\,(\omega\,(M-n))$$

where
Equation:

$$a(n) = \begin{array}{ll} 2h(n) & \text{for } \; 0 \leq n \leq M-1 \\ h(M) & \text{for } \; n = M \\ 0 & \text{otherwise} \end{array}$$

and [link] becomes
Equation:

$$A\,(\omega) \;=\; \sum_{n=0}^{M} a\,(n)\;\cos\,(\omega n)$$

with
Equation:

$$a(n) = \begin{array}{ll} h(M) & \text{for } \; n = 0 \\ 2h(M+n) & \text{for } \; 1 \leq n \leq M. \\ 0 & \text{otherwise} \end{array}$$

Notice from [link] for N odd, $A(\omega)$ is an even function around $\omega = 0$ and $\omega = \pi$, and is periodic with period 2π.

For the case where N is even,
Equation:

$$A\,(\omega) \;=\; \sum_{n=0}^{N/2-1} 2h\,(n)\,\cos\,\omega\,(M-n)$$

or with a change of variables,
Equation:

$$A\,(\omega) \;=\; \sum_{n=1}^{N/2} 2h\,(N/2-n)\,\cos\,\omega\,(n-1/2)$$

These formulas can also be made simpler by defining new coefficients so that [link] becomes
Equation:

$$A\left(\omega\right) \;=\; \sum_{n=0}^{N/2-1} a\left(n\right)\;\cos\left(\omega\left(M-n\right)\right)$$

where
Equation:

$$a(n) = \begin{cases} 2h(n) & \text{for} 0 \leq n \leq N/2-1 \\ 0 & \text{otherwise} \end{cases}$$

and [link] becomes
Equation:

$$A\left(\omega\right) \;=\; \sum_{n=1}^{N/2} a\left(n\right)\;\cos\left(\omega\left(n-1/2\right)\right)$$

with
Equation:

$$a(n) = \begin{cases} 2h(N/2-n) & \text{for} 1 \leq n \leq N/2 \\ 0 & \text{otherwise} \end{cases}$$

Notice from [link] for N even, $\mathrm{A}(\omega)$ is an even function around ω=0, an odd function around ω=π, and is periodic with period 4π. This requires $\mathrm{A}(\pi)$=0.

For the case in [link] where $K_1 = \pi/2$, an odd symmetry is required of the form
Equation:

$$h(n) \;=\; -h(N-n-1)$$

which, for N odd, gives
Equation:

$$H\left(\omega\right) \;=\; jA\left(\omega\right)e^{jM\omega}$$

with
Equation:

$$A\left(\omega\right) \;=\; \sum_{n=0}^{M-1} 2h\left(n\right)\,\sin\,\omega\left(M-n\right)$$

and for N even
Equation:

$$A\left(\omega\right) \;=\; \sum_{n=0}^{N/2-1} 2h\left(n\right)\,\sin\,\omega\left(M-n\right)$$

To calculate the frequency or amplitude response numerically, one must consider samples of the continuous frequency response function above. L samples of the general complex frequency response $H(\omega)$ in [link] are calculated from
Equation:

$$H\left(\omega_k\right) = \sum_{n=0}^{N-1} h\left(n\right) e^{-j\omega_k n}.$$

for $k = 0, 1, 2, \cdots, L-1$. This can be written with matrix notation as
Equation:

$$H = F\,h$$

where H is an L by 1 vector of the samples of the complex frequency response, F is the L by N matrix of complex exponentials from [link], and h is the N by 1 vector of real filter coefficients.

These equations are possibly redundant for equally spaced samples since $A(\omega)$ is an even function and, if the phase response is linear, $h(n)$ is symmetric. These redundancies are removed by sampling [link] over $0 \leq \omega_k \leq \pi$ and by using a defined in [link] rather than h. This can be written
Equation:

$$A = C\,a$$

where A is an L by 1 vector of the samples of the real valued amplitude frequency response, C is the L by M real matrix of cosines from [link], and a is the M by 1 vector of filter coefficients related to the impulse response by [link]. A similar set of equations can be written from [link] for N odd or from [link] for N even.

This formulation becomes a filter design method by giving the samples of a desired amplitude response as $A_d\left(k\right)$ and solving [link] for the filter coefficients $a(n)$. If the number of independent frequency samples is equal to the number of independent filter coefficients and if C is not singular, this is the frequency sampling filter design method and the frequency response of the designed filter will interpolate the specified samples. If the number of frequency samples L is larger than the number of filter coefficients N, [link] may be solved approximately by minimizing the norm $\parallel A\left(\omega\right) - A_d\left(\omega\right) \parallel$.

The Discrete Time Fourier Transform with Normalization

The discrete time Fourier transform of the impulse response of a digital filter is its frequency response, therefore, it is an important tool. When the symmetry conditions of linear phase are incorporated into the DTFT, it becomes similar to the discrete cosine or sine transform (DCT or DST). It also has an arbitrary normalization possible for the odd length that needs to be understood.

The discrete time Fourier transform (DTFT) is defined in [link] which, with the conditions of an odd length-N symmetrical signal, becomes
Equation:

$$A\left(\omega\right) = \sum_{n=1}^{M} a\left(n\right)\ \cos\left(\omega n\right) + K\,a\left(0\right)$$

where $M = (N-1)/2$. Its inverse as
Equation:

$$a(n) = \frac{2}{\pi} \int_0^{\pi} A(\omega) \cos(\omega n) \, d\omega$$

for $n = 1, 2, \cdots, M$ and
Equation:

$$a(0) = \frac{1}{K\pi} \int_0^{\pi} A(\omega) \, d\omega$$

where K is a parameter of normalization for the $a(0)$ term with $0 < K < \infty$. If $K = 1$, the expansion equation [link] is one summation and doesn't have to have the separate term for $a(0)$. If $K = 1/2$, the equation for the coefficients [link] will also calculate the $a(0)$ term and the separate equation [link] is not needed. If $K = 1/\sqrt{2}$, a symmetry results which simplifies equations later in the notes.

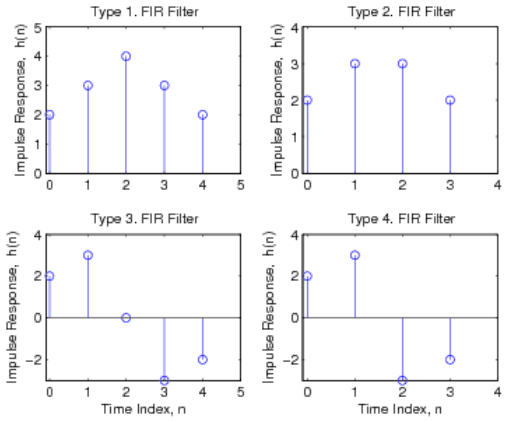
Four Types of Linear-Phase FIR Filters

From the previous discussion, it is seen that there are four possible types of FIR filters [link] that lead to the linear phase of [link]. These are summarized in [link].

Type 1.	The impulse response has an odd length and is even symmetric about its midpoint of $n = M = (N-1)/2$ which requires $h(n) = h(N-n-1)$ and gives [link] and [link].
Type 2.	The impulse response has an even length and is even symmetric about M, but M is not an integer. Therefore, there is no $h(n)$ at the point of symmetry, but it satisfies [link] and [link].
Type 3.	The impulse response has an odd length as for Type 1 and has the odd symmetry of [link], giving an imaginary multiplier for the linear-phase form in [link] with amplitude [link].
Type 4.	The impulse response has an even length as for Type 2 and the odd symmetry of Type 3 in [link] and [link] with amplitude [link].

The Four Types of Linear Phase FIR Filters

Examples of the four types of linear-phase FIR filters with the symmetries for odd and even length are shown in [link]. Note that for N odd and $h(n)$ odd symmetric, $h(M) = 0$.



Example of Impulse Responses for the Four Types of Linear Phase FIR Filters

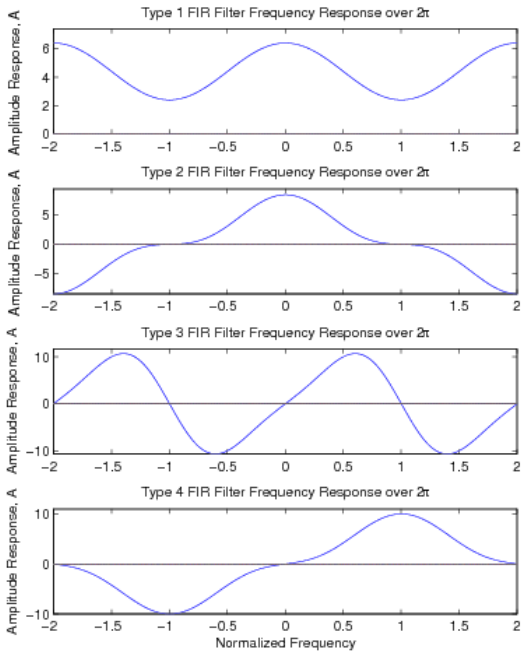
For the analysis or design of linear-phase FIR filters, it is necessary to know the characteristics of $A(\omega)$. The most important characteristics are shown in [link].

TYPE 1.	Odd length, even symmetric $h(n)$	
	$A(\omega)$ is even about $\omega = 0$	$A(\omega) = A(-\omega)$
	$A(\omega)$ is even about $\omega = \pi$	$A(\pi + \omega) = A(\pi - \omega)$
	$A(\omega)$ is periodic with period = 2π	$A(\omega + 2\pi) = A(\omega)$
TYPE 2.	Even length, even symmetric $h(n)$	
	$A(\omega)$ is even about $\omega = 0$	$A(\omega) = A(-\omega)$
	$A(\omega)$ is odd about $\omega = \pi$	$A(\pi + \omega) = -A(\pi - \omega)$
	$A(\omega)$ is periodic with period 4π	$A(\omega + 4\pi) = A(\omega)$
TYPE 3.	Odd length, odd symmetric $h(n)$	
	$A(\omega)$ is odd about $\omega = 0$	$A(\omega) = -A(-\omega)$
	$A(\omega)$ is odd about $\omega = \pi$	$A(\pi + \omega) = -A(\pi - \omega)$
	$A(\omega)$ is periodic with period $= 2\pi$	$A(\omega + 2\pi) = A(\omega)$
TYPE 4.	Even length, odd symmetric $h(n)$	

	$A(\omega)$ is odd about $\omega = 0$	$A(\omega) = -A(-\omega)$
	$A(\omega)$ is even about $\omega = \pi$	$A(\pi + \omega) = A(\pi - \omega)$
	$A(\omega)$ is periodic with period $= 4\pi$	$A(\omega + 4\pi) = A(\omega)$

Characteristics of $A(\omega)$ for Linear Phase

Examples of the amplitude function for odd and even length linear-phase filter $A(\omega)$ are shown in [link].



Example of Amplitude Responses for the Four Types of Linear Phase FIR Filters

These characteristics reveal several inherent features that are extremely important to filter design. For Types 3 and 4, $A(0) = 0$ for any choice of filter coefficients $h(n)$. This would not be desirable for a lowpass filter. Types 2 and 3 always have $A(\pi) = 0$ which is not desirable for a highpass filter. In addition to the linear-phase characteristic that represents a time shift, Types 3 and 4 give a constant 90-degree phase shift, desirable for a differentiator or Hilbert transformer. The first step in the design of a linear-phase FIR filter is the choice of the type most compatible with the specifications.

It is possible to uses the formulas to express the frequency response of a general complex or non-linear phase FIR filter by taking the even and odd parts of $h(n)$ and calculating a real and imaginary “amplitude" that would be added to give the actual frequency response.

Calculation of FIR Filter Frequency Response

As shown earlier, L equally spaced samples of $H(\omega)$ are easily calculated for $L > N$ by appending $L - N$ zeros to $h(n)$ for a length-L DFT. This appears as
Equation:

$$H(2\pi k/L) \;=\; \mathscr{DFT}\{h(n)\} \text{ for } k = 0, 1, \cdots, L-1$$

This direct method of calculation is a straightforward and flexible approach. Only the samples of $H(\omega)$ that are of interest need to be calculated. In fact, even nonuniform spacing of the frequency samples can be achieved by sampling the DTFT defined in [link]. The direct use of the DFT can be inefficient, and for linear-phase filters, it is $A(\omega)$, not $H(\omega)$, that is the most informative. In addition to the direct application of the DFT, special formulas are developed in Equation 5 from FIR Filter Design by Frequency Sampling or Interpolation for evaluating samples of $A(\omega)$ that exploit the fact that $h(n)$ is real and has certain symmetries. For long filters, even these formulas are too inefficient, so the DFT is used, but implemented by a Fast Fourier Transform (FFT) algorithm.

In the special case of Type 1 filters with L equally spaced sample points, the samples of the frequency response are of the form
Equation:

$$A_k \;=\; A\left(2\pi k/L\right) \;=\; \sum_{n=0}^{M-1} 2h\left(n\right) \cos\left(2\pi\left(M-n\right)k/L\right) + h\left(M\right)$$

For Type 2 filters,
Equation:

$$A_k \;=\; A\left(2\pi k/L\right) \;=\; \sum_{n=0}^{N/2-1} 2h\left(n\right) \cos\left(2\pi\left(M-n\right)k/L\right)$$

For Type 3 filters,
Equation:

$$A_k \;=\; A\left(2\pi k/L\right) \;=\; \sum_{n=0}^{M-1} 2h\left(n\right) \sin\left(2\pi\left(M-n\right)k/L\right)$$

For Type 4 filters,
Equation:

$$A_k \;=\; A\left(2\pi k/L\right) \;=\; \sum_{n=0}^{N-1} 2h\left(n\right) \sin\left(2\pi\left(M-n\right)k/L\right)$$

Although this section has primarily concentrated on linear-phase filters by taking their symmetries into account, the method of taking the DFT of $h(n)$ to obtain samples of the frequency response of an FIR filter also holds for general arbitrary linear phase filters.

Zero Locations for Linear-Phase FIR Filters

A qualitative understanding of the filter characteristics can be obtained from an examination of the location of the $N-1$ zeros of an FIR filter's transfer function. This transfer function is given by the z-transform of the length-N impulse response

Equation:

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

which can be rewritten as

Equation:

$$H(z) = z^{-N+1} \left(h_0 z^{N-1} + h_1 z^{N-2} + \ldots + h_{N-1} \right)$$

or as

Equation:

$$H(z) = z^{-N+1} D(z)$$

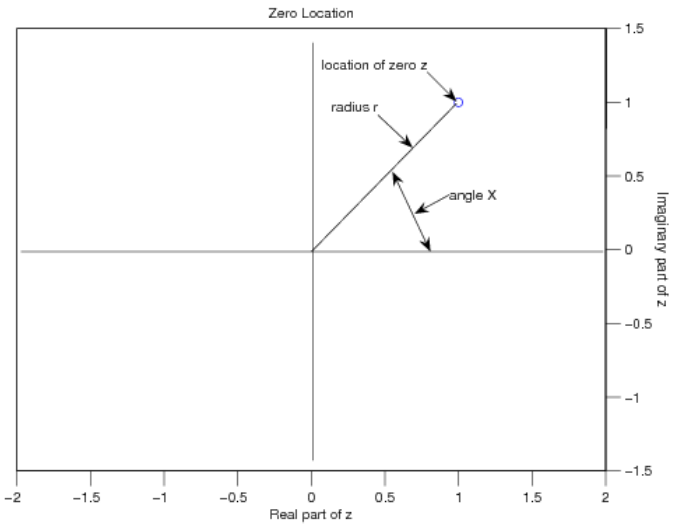
where $D(z)$ is an $N-1$ order polynomial that is multiplied by an $N-1$ order pole located at the origin of the complex z-plane. $D(z)$ is defined in order to have a simple polynomial in positive powers of z.

The fact that h(n) is real valued requires the zeros to all be real or occur in complex conjugate pairs. If the FIR filter is linear phase, there are further restrictions on the possible zero locations. From [link], it is seen that linear phase implies a symmetry in the impulse response and, therefore, in the coefficients of the polynomial $D(z)$ in [link]. Let the complex zero z_1 be expressed in polar form by

Equation:

$$z_1 = r_1 e^{jx}$$

where r_1 is the radial distance of z_1 from the origin in the complex z-plane, and x is the angle from the real axis as shown in [link].



Example of Impulse Responses for the Four Types of Linear Phase FIR Filters

Using the definition of $H(z)$ and $D(z)$ in [link] and [link] and the linear-phase even symmetry requirement of
Equation:

$$h(n) = h(N-1-n)$$

gives
Equation:

$$H(1/z) = D(z)$$

which implies that if z_1 is a zero of $H(z)$, then $1/z_1$ is also a zero of $H(z)$. In other words, if
Equation:

$$H(z_1) = 0, \quad \text{then} \quad H(1/z_1) = 0.$$

This means that if a zero exists at a radius of r_1, then one also exists at a radius of $1/r_1$, thus giving a special type of symmetry of the zeros about the unit circle. Another possibility is that the zero lies on the unit circle with $r_1 = 1/r_1 = 1$.

There are four essentially different cases [link] of even symmetric filters that have the lowest possible order. All higher order symmetric filters have transfer functions that can be factored into products of these lowest order transfer functions. These are illustrated by four basic filters of lowest order that satisfy these conditions: one length-2, two length-3, and one length-5.

The only length-2 even-symmetric linear-phase FIR filter has the form
Equation:

$$D(z) = (z+1)K$$

which, for any constant K, has a single zero at $z_1 = -1$.

The even symmetric length-3 filter has a form
Equation:

$$D(z) = \left(z^2 + az + 1\right)K$$

There are two possible cases. For $|a| > 2$, two real zeros can satisfy [link] with $z_1 = r$ and $1/r$. This gives
Equation:

$$D(z) = \left(z^2 + (r + 1/r)z + 1\right)K$$

The other length-3 case for $|a| < 2$ has two complex conjugate zeros on the unit circle and is of the form
Equation:

$$D(z) = \left(z^2 + (2\cos x)z + 1\right)K$$

The special case for $a = 2$ is not of lowest order. It can be factored into [link] squared. Any length-4 even-symmetric filter can be factored into products of terms of the form of [link] and [link].

The fourth case is of an even-symmetric length-5 filter of the form
Equation:

$$D(z) = z^4 + az^3 + bz^2 + az + 1$$

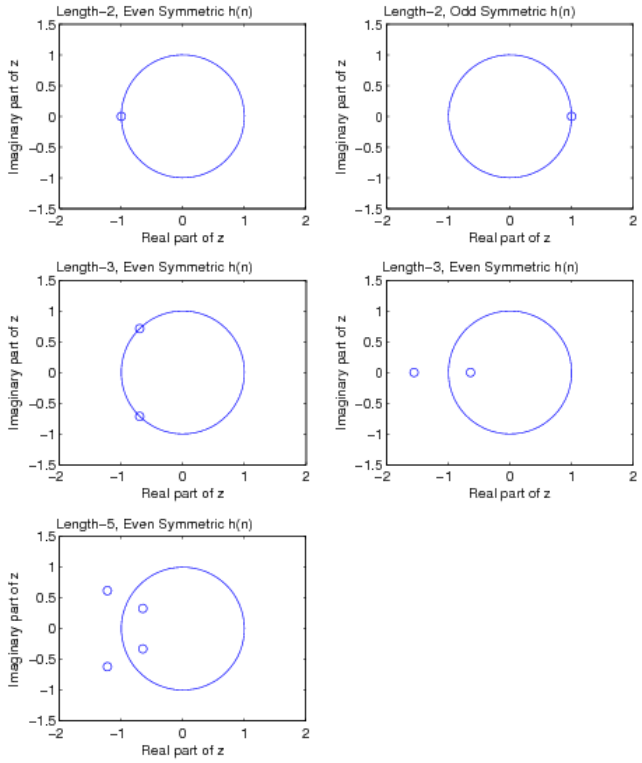
For $a^2 < 4(b-2)$ and $b > 2$, the zeros are neither real nor on the unit circle; therefore, they must have complex conjugates and have images about the unit circle. The form of the transfer function is
Equation:

$$D(z) = \left\{z^4 + \left[\left(2\left(r^2+1\right)/r\right)\cos x\right]z^3 + \left[r^2 + 1/r^2 + 4\cos^2 x\right]z^2 + \left[\left(2\left(r^2+1\right)/r\right)\cos x\right]z + 1\right\}K$$

If one of the zeros of a length-5 filter is on the real axis or on the unit circle, $D(z)$ can be factored into a product of lower order terms of the forms in [link], [link], and [link] and, therefore, is not of lowest order. The odd symmetric filters of [link] are described by the above factors plus the basic length-2 filter described by
Equation:

$$D(z) = (z-1)K$$

The zero locations for the four basic cases of Type 1 and 2 FIR filters are shown in [link]. The locations for the Type 3 and 4 odd-symmetric cases of [link] are the same, plus the zero at one from [link].



Zero Locations for the Basic Linear-Phase FIR Filter

From this analysis, it can be concluded that all linear-phase FIR filters have zeros either on the unit circle or in the reciprocal symmetry of [link] or [link] about the unit circle, and their transfer functions can always be factored into products of terms with these four basic forms. This factored form can be used in implementing a filter by cascading short filters to realize a long filter. Knowledge of the locations of the transfer function zeros helps in developing filter design and analysis programs. Notice how these zero locations are consistent with the amplitude responses illustrated in [link] and [link].

Section Summary

In this section the basic characteristics of the FIR filter have been derived. For the linear-phase case, the frequency response can be calculated very easily. The effects of the linear phase can be separated so that the amplitude can be approximated as a real-valued function. This is a very useful property for filter design. It was shown that there are four basic types of linear-phase FIR filters, each with characteristics that are also important for design. The frequency response can be calculated by application of the DFT to the filter coefficients or, for greater resolution, to the N filter coefficients with zeros added to increase the length. A very efficient calculation of the DFT uses the Fast Fourier Transform (FFT). The frequency response can also be calculated by special formulas that include the effects of linear phase.

Because of the linear-phase requirements, the zeros of the transfer function must lie on the unit circle in the z plane or occur in reciprocal pairs around the unit circle. This gives insight into the effects of the zero

locations on the frequency response and can be used in the implementation of the filter.

The FIR filter is very attractive from several points of view. It alone can achieve exactly linear phase. It is easily designed using methods that are linear. The filter cannot be unstable. The implementation or realization in hardware or on a computer is basically the calculation of an inner product, which can be accomplished very efficiently. On the negative side, the FIR filter may require a rather long length to achieve certain frequency responses. This means a large number of arithmetic operations per output value and a large number of coefficients that have to be stored. The linear-phase characteristic makes the time delay of the filter equal to half its length, which may be large.

How the FIR filter is implemented and whether it is chosen over alternatives depends strongly on the hardware or computer to be used. If an array processor is used, an FFT implementation [link] would probably be selected. If a fixed point TMS320 signal processor is used, a direct calculation of the inner product is probably best. If a floating point DSP or microprocessor with floating-point arithmetic is used, an IIR filter may be chosen over the FIR, or the implementation of the FIR might take into account the symmetries of the filter coefficients to reduce arithmetic. To make these choices, the characteristics developed in this chapter, together with the results developed later in these notes, must be considered.

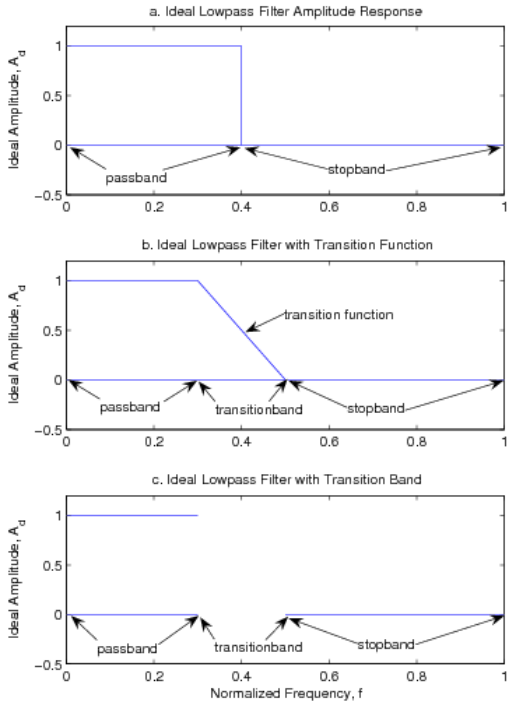
FIR Digital Filter Design

A central characteristic of engineering is design. Basic to DSP is the design of digital filters. In many cases, the specifications of a design is given in the frequency domain and the evaluation of the design is often done in the frequency domain. A typical sequence of steps in design might be:

1. From an application, choose a desired ideal response, typically described in the frequency domain.
2. From the available hardware and software, choose an allowed class of filters (e.g. a length-N FIR digital filter).
3. From the application, set a measure or criterion of “goodness" for the response of an allowed filter compared to the desired response.
4. Develop a method to find (or directly generate) the best member of the allowed class of linear phase FIR filters as measured by the criterion of goodness.

This approach is often used iteratively. After the best filter is designed and evaluated, the desired response and/or the allowed class and/or the measure of quality might be changed; then the filter would be redesigned and reevaluated.

The ideal response of a lowpass filter is given in [link].



Ideal Amplitude Responses of Linear Phase FIR Filters

[link]a is the basic lowpass response that exactly passes frequencies from zero up to a certain frequency, then rejects (multiplies those frequency components by zero)the frequencies above that. [link]b introduces a “transitionband" between the pass and stopband to make the design easier and more efficient. [link]c introducess a transitionband which is not used in the approximation of the actual to the ideal responses. Each of these ideal responses (or other similar ones) will fit a particular application best.

FIR Filter Design by Frequency Sampling or Interpolation

Since samples of the frequency response of an FIR filter can be calculated by taking the DFT of the impulse response $h(n)$, one could propose a filter design method consisting of taking the inverse DFT of samples of a desired frequency response. This can indeed be done and is called **frequency sampling design**. The resulting filter has a frequency response that exactly interpolates the given samples, but there is no explicit control of the behavior between the samples [link], [link].

Three methods for frequency sampling design are:

1. Take the inverse DFT (perhaps using the FFT) of equally spaced samples of the desired frequency response. Care must be taken to use the correct phase response to obtain a real valued causal $h(n)$ with reasonable behavior between sample response. This method works for general nonlinear phase design as well as for linear phase.
2. Derive formulas for the inverse DFT which take the symmetries, phase, and causality into account. It is interesting to notice these analysis and design formulas turn out to be the discrete cosine and sine transforms and their inverses.
3. Solve the set of simultaneous linear equations that result from calculating the sampled frequency response from the impulse response. This method allows unevenly spaced samples of the desired frequency response but the resulting equations may be ill conditioned.

Frequency Sampling Filter Design by Inverse DFT

The most direct frequency sampling design method is to simply take the inverse DFT of equally spaced samples of the desired complex frequency response $H_d(\omega_k)$. This is done by
Equation:

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} H_d\left(\frac{2\pi}{N}k\right) e^{j2\pi nk/N}$$

where care must be taken to insure that the real and imaginary parts (or magnitude and phase) of $H_d(\omega_k)$ satisfy the symmetry conditions that give a real, causal $h(n)$. This method will allow a general complex $H(\omega)$ as well as a linear phase. In most cases, it is easier to specify proper and consistent samples if it is the magnitude and phase that are set rather than the real and imaginary parts. For example, it is important that the desired phase be consistent with the specified length being even or odd as is given in Equation 28 from FIR Digital Filters and Equation 24 from FIR Digital Filters.

Since the frequency sampling design method will always produce a filter with a frequency response that interpolates the specified samples, the results of inappropriate phase specifications will show up as undesired behavior between the samples.

Frequency Sampling Filter Design by Formulas

When equally spaced samples of the desired frequency response are used, it is possible to derive formulas for the inverse DFT and, therefore, for the filter coefficients. This is because of the orthogonal basis function of the DFT. These formulas can incorporate the various constraints of a real $h(n)$ and/or linear phase and eliminate the problems of inconsistency in specifying $H(\omega_k)$.

To develop explicit formulas for frequency-sampling design of linear-phase FIR filters, a direct use of the inverse DFT is most straightforward. When $H(\omega)$ has linear phase, [link] may be simplified using the formulas for the four types of linear-phase FIR filters.

Type 1. Odd Sampling

Samples of the frequency response Equation 29 from FIR Digital Filters for the filter where N is odd, $L = N$, and $M = (N-1)/2$, and where there is a frequency sample at $\omega = 0$ is given as
Equation:

$$A_k = \sum_{n=0}^{M-1} 2h(n) \cos\left(2\pi(M-n)k/N\right) + h(M).$$

Using the amplitude function $A(\omega)$, defined in Equation 28 from FIR Digital Filters, of the form [link] and the IDFT [link] gives for the impulse response
Equation:

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{-j2\pi Mk/N} A_k e^{j2\pi nk/N}$$

or
Equation:

$$h(n) = \frac{1}{N} \sum_{k=0}^{N-1} A_k e^{j2\pi(n-M)k/N}.$$

Because $h(n)$ is real, $A_k = A_{N-k}$ and [link] becomes
Equation:

$$h(n) = \frac{1}{N}\left[A_0 + \sum_{k=1}^{M-1} 2A_k \cos\left(2\pi(n-M)k/N\right)\right].$$

Only $M+1$ of the $h(n)$ need be calculated because of the symmetries in Equation 27 from FIR Digital Filters.

This formula calculates the impulse response values $h(n)$ from the desired frequency samples A_k and requires M^2 operations rather than N^2. An interesting observation is that not only are [link] and [link] a pair of analysis and design formulas, they are also a transform pair. Indeed, they are of the same form as a discrete cosine transform (DCT).

Type 2. Odd Sampling

A similar development applied to the cases for even N from Equation 36 from FIR Digital Filters gives the amplitude frequency response samples as
Equation:

$$A_k \;=\; \sum_{n=0}^{N/2-1} 2h\,(n)\,\cos\,(2\pi\,(M-n)k/N)$$

with the design formula of
Equation:

$$h\,(n) \;=\; \frac{1}{N}\left[A_0 + \sum_{k=1}^{N/2-1} 2A_k\,\cos\,(2\pi\,(n-M)k/N)\right]$$

which is of the same form as [link], except that the upper limit on the summation recognizes N as even and $A_{N/2}$ equals zero.

Even Sampling

The schemes just described use frequency samples at
Equation:

$$\omega \;=\; 2\pi k/N, \qquad k = 0, 1, 2, \ldots, N-1$$

which are N equally-spaced samples starting at $\omega = 0$. Another possible pattern for frequency sampling that allows design formulas has no sample at $\omega = 0$, but uses N equally-spaced samples located at
Equation:

$$\omega \;=\; (2k+1)\pi/N, \qquad k = 0, 1, 2, \ldots, N-1$$

This form of frequency sampling is more difficult to relate to the DFT than the sampling of [link], but it can be done by stretching A_k and taking a 2N-length DFT [link].

Type 1. Even Sampling

The two cases for odd and even lengths and the two for samples at zero and not at zero frequency give a total of four cases for the frequency-sampling design method applied to linear- phase FIR filters of Types 1 and 2, as defined in the section Linear-Phase FIR Filters. For the case of an odd length and no zero sample, the analysis and design formulas are derived in a way analogous to [link] and [link] to give
Equation:

$$A_k \;=\; \sum_{n=0}^{M-1} 2h\,(n)\,\cos\,(2\pi\,(M-n)\,(k+1/2)/N) + h\,(M)$$

The design formula becomes
Equation:

$$h\,(n) \;=\; \frac{1}{N}\left[\sum_{k=0}^{M-1} 2A_k\,\cos\,(2\pi\,(n-M)\,(k+1/2)/N) + A_M\,\cos\,\pi\,(n-M)\right]$$

Type 2. Even Sampling

The fourth case, for an even length and no zero frequency sample, gives the analysis formula
Equation:

$$A_k \;=\; \sum_{n=0}^{N/2-1} 2h\,(n)\,\cos\,(2\pi\,(M-n)\,(k+1/2)/N)$$

and the design formula
Equation:

$$h\,(n) \;=\; \frac{1}{N}\left[\sum_{k=0}^{N/2-1} 2A_k\,\cos\,(2\pi\,(n-M)\,(k+1/2)/N)\right]$$

These formulas in [link], [link], [link], and [link] allow a very straightforward design of the four frequency-sampling cases. They and their analysis companions in [link], [link], [link], and [link] also are the four forms of discrete cosine and inverse-cosine transforms. Matlab programs which implement these four designs are given in the appendix.

Type 3. Odd Sampling

The design of even-symmetric linear-phase FIR filters of Types 1 and 2 in the section Linear-Phase FIR Filters have been developed here. A similar development for the odd-symmetric filters, Types 3 and 4, can easily be performed with the results closely related to the discrete sine transform. The Type 3 analysis and design results using the frequency sampling scheme of [link] are
Equation:

$$A_k \;=\; \sum_{b=0}^{M-1} 2h\,(n)\,\sin\,(2\pi\,(M-n)k/N)$$

and

Equation:

$$h\left(n\right) \;=\; \frac{1}{N}\left[\sum_{k=1}^{M} 2A_k \,\sin\left(2\pi\left(M-n\right)k/N\right)\right]$$

Type 4. Odd Sampling

For Type 4 they are
Equation:

$$A_k \;=\; \sum_{n=0}^{N/2-1} 2h\left(n\right) \sin\left(2\pi\left(M-n\right)k/N\right)$$

and
Equation:

$$h\left(n\right) \;=\; \frac{1}{N}\left[\sum_{k=1}^{N/2-1} 2A_k \,\sin\left(2\pi\left(M-n\right)k/N\right) + A_{N/2}\,\sin\left(\pi\left(M-n\right)\right)\right].$$

Type 3. Even Sampling

Using the frequency sampling scheme of [link], the Type 3 equations become
Equation:

$$A_k \;=\; \sum_{n=0}^{M-1} 2h\left(n\right) \sin\left(2\pi\left(M-n\right)\left(k+1/2\right)/N\right)$$

and
Equation:

$$h\left(n\right) \;=\; \frac{1}{N}\left[\sum_{k=0}^{M-1} 2A_k \,\sin\left(2\pi\left(M-n\right)\left(k+1/2\right)/N\right)\right]$$

Type 4. Even Sampling

For Type 4 they are
Equation:

$$A_k \;=\; \sum_{n=0}^{N/2-1} 2h\,(n)\,\sin\,(2\pi\,(M-n)\,(k+1/2)/N)$$

and
Equation:

$$h\,(n) \;=\; \frac{1}{N}\left[\sum_{k=0}^{N/2-1} 2A_k\,\sin\,(2\pi\,(M-n)\,(k+1/2)/N)\right].$$

These Type 3 and 4 formulas are useful in the design of differentiators and Hilbert transformers [1,2,9,31] directly and as the base of the discrete least-squared-error methods in the section Discrete Frequency Samples of Error.

Frequency Sampling Design of FIR Filters by Solution of Simultaneous Equations

A direct way of designing FIR filters from samples of a desired amplitude simply takes the sampled definition of the frequency response Equation 29 from FIR Digital Filters as
Equation:

$$A\,(\omega_k) \;=\; \sum_{n=0}^{M-1} 2h\,(n)\,\cos\,\omega_k\,(M-n) + h\,(M)$$

or the reduced form from Equation 37 from FIR Digital Filters as
Equation:

$$A\,(\omega_k) \;=\; \sum_{n=0}^{M} a\,(n)\;\cos\,(\omega_k\,(M-n))$$

where
Equation:

$$a(n) = \begin{cases} 2h(n) & \text{for } \; 0 \le n \le M-1 \\ h(M) & \text{for } \; n = M \\ 0 & \text{otherwise} \end{cases}$$

for k=0,1,2,...,M and solves the M+1 simultaneous equations for a(n) or equivalently, h(n). Indeed, this approach can be taken with general non-linear phase design from

Indeed, this approach can be taken with general non-linear phase design from
Equation:

$$H(\omega_k) = \sum_{n=0}^{N-1} h(n) e^{j\omega_k n}$$

for $k = 0, 1, 2, \cdots, N-1$ which gives N equations with N unknowns.

This design by solving simultaneous equations allows non-equally spaced samples of the desired response. The disadvantage comes from the numerical calculations taking considerable time and being subject to inaccuracies if the equations are ill-conditioned.

The frequency sampling design method is interesting but is seldom used for direct design of filters. It is sometimes used as an interpolating method in other design procedures to find $h(n)$ from calculated $A(\omega_k)$. It is also used as a basis for a least squares design method discussed in the next section.

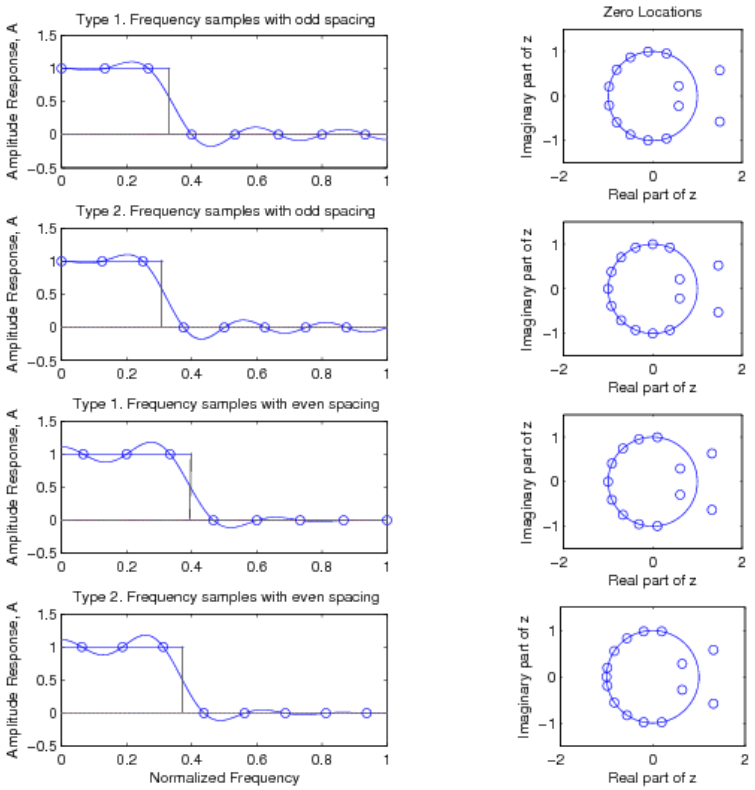
Examples of Frequency Sampling FIR Filter Design

To show some of the characteristics of FIR filters designed by frequency sampling, we will design a Type 1., length-15 FIR low pass filter. Desired amplitude response was one in the pass band and zero in the stop band. The cutoff frequency was set at approximately $f = 0.35$ normalized. Using the formulas [link], [link], [link], and [link], we got impulse responses $h(n)$, which are use to generate the results shown in Figures [link] and [link].

The Type 1, length-15 filter impulse response is:
Equation:

$$h1 = -0.5 \quad 0 \quad 1.1099 \quad 0 \quad -1.6039 \quad 0 \quad 4.494 \quad 7 \quad 4.494 \quad 0 \quad -1.6039 \quad 0 \quad 1.1099 \quad 0 \quad -0.5$$

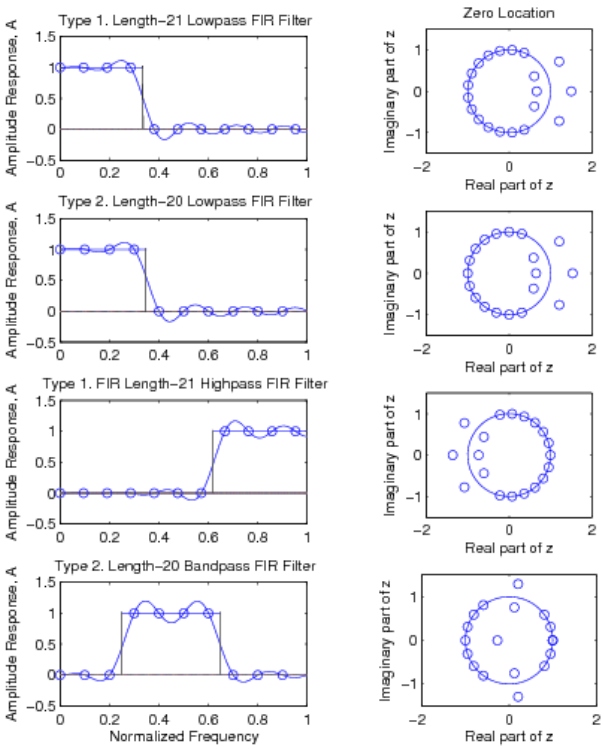
The amplitude frequency response and zero locations are shown in [link]a



Frequency Responses and Zero Locations of Length-15 and 16 FIR Filters
Designed by Frequency Sampling

We see a good lowpass filter frequency response with the actual amplitude interpolating the desired values at 8 equally spaced points. Notice there is considerable overshoot near the cutoff frequency. This is characteristic of frequency sampling designs and is a sort of “Gibbs phenomenon" but is even worse than that in a Fourier series expansion of a discontinuity. This Gibbs phenomenon could be reduced by using unequally spaced samples and designing by solving simultaneous equations. Imagine sampling in the pass and stop bands of Figure 8c from FIR Digital Filters but not in the transitionband. The other responses and zero locations show the results of different interpolation locations and lengths. Note the zero at -1 for the even filters.

Examples of longer filters and of highpass and bandpass frequency sampling designs are shown in [link]. Note the difference of even and odd distributions of samples with with or without an interpolation point at zero frequency. Note the results of different ideal filters and Type 1 or 2. Also note the relationship of the amplitude response and zero locations.



Frequency Response and Zero Locations of FIR Filters Designed by Frequency Sampling

Least Squared Error Design of FIR Filters

Because the integral of the square of a signal is a measure of its energy, there is some physical reason for minimizing the integral of the squared error [link], [link]. Also, because of Parseval's theorem, a least squares approximation in the frequency domain is a least squares approximation in the time domain. However, minimizing the **worst case** squared error induces a minimum Chebyshev error problem in some formulations [link].

Discrete Frequency Samples of Error

If we approximate the integral squared error by the sum of the squared error as given by
Equation:

$$\begin{aligned} q &= \frac{1}{L}\sum_{k=0}^{L-1}\left(A\left(\omega_k\right)-A_d\left(\omega_k\right)\right)^2 = \frac{1}{L}\sum_{k=0}^{L-1} e(\omega_k)^2 \\ &\approx \int_0^{\pi}\left(A\left(\omega\right)-A_d\left(\omega\right)\right)^2 d\omega = \int_0^{\pi} e(\omega)^2\, d\omega \end{aligned}$$

where the approximation error as a function of frequency is defined by $e\left(\omega\right)=A\left(\omega\right)-A_d\left(\omega\right)$ with $A(\omega)$ being the amplitude response of the filter and $A_d\left(\omega\right)$ being the desired amplitude response or the ideal response. The matrix statement for the error vector becomes
Equation:

$$\epsilon = \mathbf{A}-\mathbf{A_d} = \mathbf{C}\,\mathbf{a}-\mathbf{A_d}$$

where $\mathbf{C}$ is the matrix of cosines from Equation 48 from FIR Digital Filters, $\mathbf{a}$ is the vector of half of the filter coefficients from Equation 48 from FIR Digital Filters, and $\mathbf{A_d}$ is the vector of samples of the ideal desired amplitude response. The number of samples of the amplitude response is L which should be five to twenty times the length of the filter to give a good approximation of the integral in most cases. The error to be minimized is
Equation:

$$q = \epsilon^{\mathbf{T}}\epsilon$$

except for a scale factor of $\frac{1}{L}$.

This could also be posed for the general phase problem by using $H(\omega_k)$ rather than $A(\omega_k)$ and $h(n)$, the actual impulse response, rather than $a(n)$, a nomralized half of the impulse response.

Truncated frequency sampling design using the inverse FFT or IDCT

The design problem is posed by defining an error measure q as a sum of the squared differences between the actual and the desired frequency response over a set of L frequency samples. This error function is defined as
Equation:

$$q = \frac{1}{L}\sum_{k=0}^{L-1}\left|H\left(\omega_k\right) - H_d\left(\omega_k\right)\right|^2$$

where $H_d\left(\omega_k\right)$ are the L samples of the desired response. This problem is easier to formulate and solve if the frequency samples are equally spaced as in Equation 8 from FIR Filter Design by Frequency Sampling or Interpolation which gives
Equation:

$$\omega_k = 2\pi k/L$$

and the problem is restricted to linear-phase filters where the real-valued amplitude $A(\omega)$ can be approximated rather than the complex frequency response $H(\omega)$. For approximations to a complex response, see "Complex and Minimum Phase Approximation".

Linear phase and equally spaced samples cause [link] to become
Equation:

$$q = \frac{1}{L}\sum_{k=0}^{L-1}\left|A\left(2\pi k/L\right) - A_d\left(2\pi k/L\right)\right|^2$$

or with a simpler notation
Equation:

$$q = \frac{1}{L}\sum_{k=0}^{L-1}\left|A_k - A_{dk}\right|^2$$

A very powerful property of the Fourier transform allows a straightforward design of least-squared-error FIR filters. Parseval's Theorem, which is based on the orthogonality of the DFT, states that the error defined by [link] in the frequency domain can also be calculated in the time domain by
Equation:

$$q = \sum_{n=0}^{L-1}\left|h\left(n\right) - h_d\left(n\right)\right|^2$$

where $h_d\left(n\right)$ is the length-L symmetric FIR filter that has the L frequency response amplitude samples A_{dk}. This may be calculated by the frequency sampling method in the section Four Types of Linear-Phase FIR Filters using the special formulas such as Equation 8 from FIR Filter Design by Frequency Sampling or Interpolation for length L or the inverse DFT. The filter to be designed has a length-N symmetric impulse response $h(n)$ with L frequency response samples A_k.

Because the filter $h(n)$ is of length-N and symmetric, the error equation [link] can be split into two sums
Equation:

$$q = \sum_{n=-M}^{M} \left| \widehat{h}(n) - \widehat{h}_d(n) \right|^2 + 2 \sum_{n=M+1}^{(L-1)/2} \left| \widehat{h}_d(n) \right|^2$$

where $\widehat{h}(n)$ and $\widehat{h}_d(n)$ are the inverse DTFTs of A_k and A_{dk} respectively, which means they are the $h(n)$ and $h_d(n)$ shifted to be symmetic about $n = 0$. This requires the number of frequency samples L must be odd.

[link] clearly shows that to minimize q, the N values of $h(n)$ are chosen to be equal to the equivalent N values of $h_d(n)$ making the first sum equal zero. In other words, $h(n)$ is obtained by symmetrically truncating $h_d(n)$. The residual error is then given by the second summation above. An examination of the residual error as a function of N may aid in the choice of the filter length N.

For the Type 1 linear-phase FIR filter (described in the section Four Types of Linear-Phase FIR Filters) which has an odd length N and an even-symmetric impulse response, the L equally spaced samples of the frequency response from Equation ? from Fir Digital Filters gives
Equation:

$$A_k = \sum_{n=0}^{M-1} 2h(n) \cos\left(2\pi(M-n)k/L\right) + h(M)$$

for $k = 0, 1, 2, \ldots, L-1$, where $M = (N-1)/2$. This formula was derived as a special case of the DFT applied to the Type 1 real, even-symmetric FIR filter coefficients to calculate the sampled amplitude of the frequency response (perhaps better posed using $a(n)$). It was noted in the section Frequency Sampling Filter Design by Formulas that it is also a cosine transform and it can be shown that this transformation is orthogonal over the independent values of A_k, just as the DFT is.

The desired ideal amplitude gives the ideal impulse response $h_d(n)$ from Equation 29 from FIR Digital Filters by
Equation:

$$h_d(n) = \frac{1}{N}\left[A_{d0} + \sum_{k=1}^{M-1} 2A_{dk} \cos\left(2\pi(n-M)k/N\right)\right].$$

for $n = 0, 1, \cdots, L-1$. This is used in [link], and is the ideal impulse response that is truncated and shifted to give a causal, symmetric $h(n)$.

Use of the alternative equally-spaced sampling in Equation 9 from FIR Filter Design by Frequency Sampling or Interpolation, which has no sample at zero frequency, requires $h_d(n)$ be calculated from Equation 11 from FIR Filter Design by Frequency Sampling or Interpolation and Equation 13 from FIR Filter Design by Frequency Sampling or Interpolation. The Type 2 filters with even N are developed in a similar way and use the design formulas Equation 36 from FIR Digital Filters and Equation 37 from FIR Digital Filters. These methods are summarized by:

"The filter design procedure for an odd-length Type 1 filter is to first design an odd-length-L FIR filter by the frequency sampling method from Equation 5 from FIR Filter Design by Frequency Sampling or Interpolation or Equation 11 from FIR Filter Design by Frequency Sampling or Interpolation or the IDFT, then to symmetrically truncate it to the desired odd-length N and shift it to make $h(n)$ causal. To design

an even-length Type 2 filter , start with an even-length-L frequency-sampling design from Equation 7 from FIR Filter Design by Frequency Sampling or Interpolation or Equation 13 from FIR Filter Design by Frequency Sampling or Interpolation or the IDFT and symmetrically truncate and shift. The resulting length-N FIR filters are optimal LS-error approximations to the desired frequency response over the L frequency samples."

This approach can also be applied to the general arbitrary phase FIR filter design problem.

Weighted, Unevenly Sampled Discrete Least Squared Error Filter Design by Solving Simultaneous Equations

It is sometimes desirable to formulate the least squared error design problem using unequally-spaced frequency samples and/or a weighting function on the error. This is not possible using the IDFT or derived formulas above and requires a different approach to the solution.

Samples of the amplitude response derived for N odd in Equation 2 from FIR Filter Design by Frequency Sampling or Interpolation are given by
Equation:

$$A\left(\omega_k\right) \; = \; \sum_{n=1}^{M} 2h\left(M-n\right) \; \cos\left(\omega_k n\right) + h\left(M\right)$$

for $k = 0, 1, \cdots, L-1$. This relates the L frequency samples $A(\omega_k)$ to the M+1 independent values of the symmetric length-N impulse response h(n). In the design problem where the A_k are given and the values for h(n) are to be found, this represents L equations with M+1 unknowns. Because of the symmetries of $A(\omega)$ shown in Figure 5 from FIR Digital Filters, only half of the L values of A_k are independent; however, in some cases, to have proper weights on all L samples, all must be calculated.

[link] sampled at L arbitrary frequencies can be written as a matrix equation
Equation:

$$\mathbf{Ca} \; = \; \mathbf{A}$$

where a is an $M+1$ length vector with elements which are the first half of $h(n)$. $\mathbf{C}$ is an L by $(M+1)$ matrix of the cosine terms from [link], and A is a length-L vector of the frequency samples $A(\omega_k)$.

If the formula for the calculation of L values of the frequency response of a length-N FIR filter in [link] is used to define an error vector of differences as defined in [link] and the result is written in the matrix formulation of Equation 48 from FIR Filter Design by Frequency Sampling or Interpolation, the error becomes
Equation:

$$\mathbf{Ca} \; = \; \mathbf{A} \; = \; \mathbf{A_d} + \mathbf{e}$$

or
Equation:

$$\mathbf{Ca} - \mathbf{A_d} \; = \; \mathbf{e}$$

where $\mathbf{e}$ is a vector of differences between the actual and desired samples of the frequency response. The error measure defined in [link] becomes the quadratic form
Equation:

$$q = \mathbf{e^T e}$$

For $L > N$, equation [link] is over determined and cannot, in general, be solved for $\mathbf{a}$. The filter design error measure is the norm of $\mathbf{e}$, as given in [link]. This error measure is minimized by making $\mathbf{e}$ orthogonal to the columns of $\mathbf{C}$ in [link]. Multiplying both sides of [link] by the transpose of $\mathbf{C}$ gives
Equation:

$$\mathbf{C^T C a} = \mathbf{C^T A_d} + \mathbf{C^T e}$$

In order for q to be minimum, $\mathbf{e}$ must be orthogonal to the columns of $\mathbf{C}$ and, therefore, $\mathbf{C^T e}$ must be zero. Hence, the optimal $\mathbf{a}$ must satisfy the “normal equations" [link], [link], [link] which are
Equation:

$$\mathbf{C^T C a} = \mathbf{C^T A_d}$$

and which can be rewritten in terms of the pseudo-inverse [link], [link] as
Equation:

$$\mathbf{a} = \left[\mathbf{C^T C}\right]^{-1} \mathbf{C^T A_d}$$

If $L = N$, this becomes the regular frequency-sampling problem and can be solved with zero error. For the case of interest in this section, where $L > N$, there are still only $M + 1$ equations to be solved. For $L >> N$, equation [link] may be ill-conditioned, and [link] should not be used to solve them. Special methods will be necessary to avoid serious numerical problems [link].

If a weighted error function is desired, [link] is modified to give
Equation:

$$q = \frac{1}{L} \sum_{k=0}^{L-1} W_k \left| A\left(\omega_k\right) - A_d\left(\omega_k\right) \right|^2$$

The normal equations of [link] become
Equation:

$$\mathbf{C^T W C a} = \mathbf{C^T W A_d}$$

where W is a positive-definite matrix of the weights. If zero weights are desired, the effect is be achieved by removing those frequencies from the set of L frequencies, not by using a zero value weight which would violate the vector-space conditions of a well-posed minimization problem.

Although developed here for the linear-phase filter, [link] is a very general design approach for the FIR filter that allows arbitrary phase, as well as uneven frequency sampling and a weighting function in the

error definition. For the arbitrary phase case, a complex $\mathbf{F}$ is obtained from sampling Equation 28 from FIR Digital Filters and the full $h(n)$ is used. For the special case of the equally-spaced frequency samples and linear- phase filter with unity weighting, the solution of [link] or [link] is the same as given by the frequency sampling design formulas.

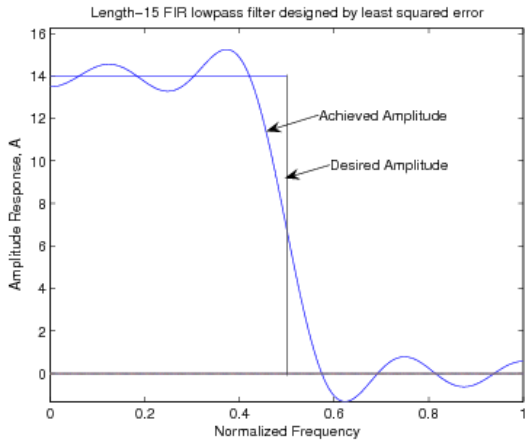
One of the important uses of the unequally spaced frequency samples is to create a transition band between the pass and stopbands where there are no samples. This “don't care" band does not contribute to the error measure q and allows better approximation to occur over the pass and stopbands.

Of the many ways to solve [link] or [link], one of the easiest and most reliable is the use of Matlab , which has a special command to solve this least-mean-squared error problem. Equation [link] should not be solved directly. For large L, it is ill-conditioned and a direct solution will probably have large errors. Matlab uses special algorithms to minimize these numerical errors.

This approach was applied to the same problems that were solved by frequency sampling in the previous section. For $N = L$, the same results are obtained, thus verifying the theoretical prediction. As L becomes larger compared to N, more control is exerted over the behavior between the original sample points. As L becomes large compared to N, the solution approaches the same results as obtained where the error is defined as a continuous function of frequency and the integral of the squared error is minimized. Although the solution of the normal equations is a powerful and flexible technique, it can be slow, have numerical problems, and require large amounts of computer memory.

Examples of Discrete Least Squared Error Filter Design

Here we will give examples of several least squared error designs of FIR filters.

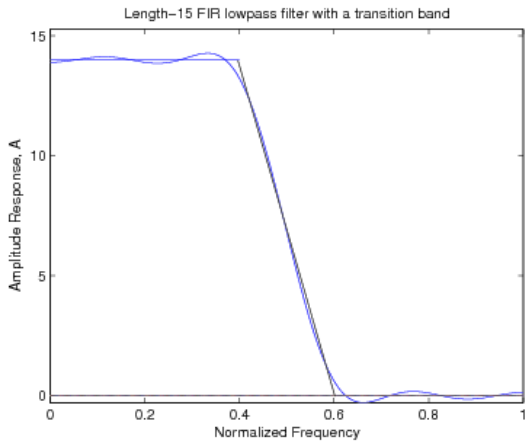


Frequency Response of Length-15 FIR Filter Designed by Least Squared Error

As for the frequency sampling design, we see a good lowpass filter frequency response with the actual amplitude interpolating the desired values at different points from the frequency sampling example in

[link] even though the length and band edge are the same. Notice there is less over shoot but more ripple near $f = 0$. The Gibbs phenomenon is the same as for the Fourier series.

If a transition band is introduces in the ideal amplitude response between $f = 0.4$ and $f = 0.6$ with a straight line, the overshoot is reduced significantly but with a slightly slower transition from the pass to stop band. This is illustrated in [link].



Frequency Response of Length-15 FIR Filter with a Transition Band Designed by Least Squared Error

Continuous Frequency Definition of Error

Because the energy of a signal is the integral of the sum of the squares of the Fourier transform magnitude and because specifications are usually given in the frequency domain, a very reasonable error measure to minimize is the integral squared error given by
Equation:

$$q = \frac{1}{\pi}\int_0^{\pi} |A_d(\omega) - A(\omega)|^2 \, d\omega$$

where $A_d(\omega)$ is the desired ideal amplitude response, $A(\omega) = \sum_n a(n) \cos(\omega(M-N)n)$ is the achieved amplitude response with the length $h(n)$ related to $h(n)$ by Equation 29 from FIR Digital Filters. This integral squared error is approximated by the discrete squared error defined in [link] for $L >> N$ which in some cases is much easier to minimize. However for some very useful cases, formulas can be found for $h(n)$ that minimize [link] and that is what we will be considering in this section.

The Unweighted Least Integral Squared Error Approximation

If the error measure is the unweighted integral squared error defined in [link], Parseval's theorem gives the equivalent time-domain formulation for the error to be
Equation:

$$q = \sum_{n=-\infty}^{\infty} \left| h_d(n) - h(n) \right|^2 = \frac{1}{\pi} \int_0^{\pi} \left| A_d(\omega) - A(\omega) \right|^2 d\omega.$$

In general, this ideal response is infinite in duration and, therefore, cannot be realized exactly by an actual FIR filter.

As was done in the case of the discrete error measure, we break the infinite sum in [link] into two parts, one of which depends on $h(n)$ and the other does not.
Equation:

$$q = \sum_{n=-M}^{M} \left| h_d(n) - h(n) \right|^2 + 2 \sum_{n=M+1}^{\infty} \left| h_d(n) \right|^2$$

Again, we see that the minimum q is achieved by using $h(n) = h_d(n)$ for $-M \leq n \leq M$. In other words, the infinitely long $h_d(n)$ is symmetrically truncated to give the optimal least integral squared error approximation. The problem then becomes one of finding the $h_d(n)$ to truncate.

Here the integral definition of approximation error is used. This is usually what we really want, but in some cases the integrals can not be carried out and the sampled method above must be used.

Ideal Constant Gain Passband Lowpass Filter

Here we assume the simplest ideal lowpass single band FIR filter to have unity passband gain for $0 < \omega < \omega_0$ and zero stopband gain for $\omega_0 < \omega < \pi$ similar to those in Figure 8a from FIR Digital Filters and [link]. This gives
Equation:

$$A_d(\omega) = \begin{cases} 1 & 0 < \omega < \omega_0 \\ 0 & \omega_0 < \omega < \pi \end{cases}$$

as the ideal desired amplitude response. The ideal shifted filter coefficients are the inverse DTFT from Equation 15 from Chebyshev or Equal Ripple Error Approximation Filters of this amplitude which for N odd are given by
Equation:

$$\widehat{h}_d(n) = \frac{1}{\pi} \int_0^{\pi} A_d(\omega) \cos(\omega n) \, d\omega$$

Equation:

$$= \frac{1}{\pi} \int_0^{\omega_0} \cos(\omega n) \, d\omega = \left(\frac{\omega_0}{\pi} \right) \frac{\sin(\omega_0 n)}{\omega_0 n}$$

which is sometimes called a “sinc" function. Note $\widehat{h}_d\,(n)$ is generally infinite in length. This is now symmetrically truncated and shifted by $M = (N-1)/2$ to give the optimal, causal length-N FIR filter coefficients as

Equation:

$$h\,(n) \;=\; \left(\frac{\omega_0}{\pi}\right)\frac{\sin\left(\omega_0\,(n-M)\right)}{\omega_0\,(n-M)} \qquad \text{for } 0 \le n \le N-1$$

and $h(n) = 0$ otherwise. The corresponding derivation for an even length starts with the inverse DTFT in Equation 5 from Constrained Approximation and Mixed Criteria for a shifted even length filter is

Equation:

$$\widehat{h}_d \;=\; \frac{1}{\pi}\int_0^{\pi} A_d\,(\omega)\;\cos\left(\omega\,(n+1/2)\right)d\omega \;=\; \left(\frac{\omega_0}{\pi}\right)\frac{\sin\left(\omega_0\,(n+1/2)\right)}{\omega_0\,(n+1/2)}$$

which when truncated and shifted by $N/2$ gives the same formula as for the odd length design in [link] but one should note that $M = (N-1)/2$ is not an integer for an even N.

Ideal Linearly Increasing Gain Passband Lowpass Filter

We now derive the design formula for a filter with an ideal amplitude response that is a linearly increasing function in the passband rather than a constant as was assumed above. This ideal amplitude response is given by and illustrated in [link] For N odd, the ideal infinitely long shifted filter coefficients are the inverse DTFT of this amplitude given by

Equation:

$$A_d(\omega) = \begin{cases} \frac{1}{\pi}\omega & 0 < \omega < \omega_0 \\ 0 & \omega_0 < \omega < \pi \end{cases}$$

and illustrated in [link] For N odd, the ideal infinitely and shifted filter coefficients are the inverse DTFT of this amplitude given by

Equation:

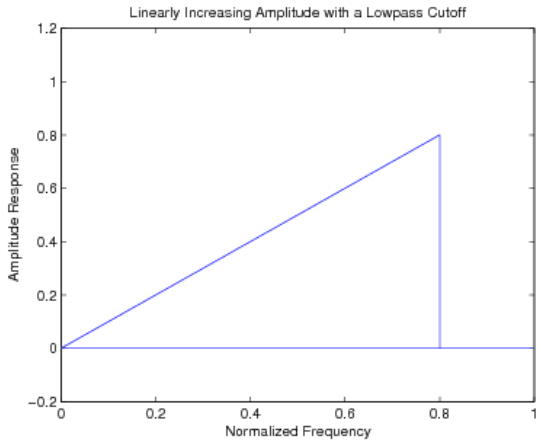
$$\widehat{h}_d\,(n) \;=\; \frac{1}{\pi}\int_0^{\omega_0}\left(\frac{\omega}{\pi}\right)\;\cos\left(\omega n\right)d\omega = \frac{\cos\left(\omega_0 n\right)-1}{\pi^2 n^2} + \frac{\omega_0\,\sin\left(\omega_0 n\right)}{\pi^2 n}$$

with the indeterminate $\widehat{h}_d\,(0) = \frac{\omega_0^2}{2\pi^2}$. This is now truncated and shifted by $M = (N-1)/2$ to give the optimal, causal length-N FIR filter coefficients as

Equation:

$$h\,(n) \;=\; \frac{\cos\left(\omega_0\,(n-M)\right)-1}{\pi^2(n-M)^2} + \frac{\omega_0\,\sin\left(\omega_0\,(n-M)\right)}{\pi^2\,(n-M)} \qquad \text{for } 0 \le n \le N-1$$

and $h(n) = 0$ otherwise. The corresponding derivation for an even length starts with the inverse DTFT for a shifted even length filter in Equation 15 from Chebyshev or Equal Ripple Error Approximation Filters and after shifting by $N/2$ gives the same result as [link].



Ideal Frequency Response of an FIR Filter with Increasing Gain in the Passband and Lowpass Cutoff

Ideal Differentiator plus Lowpass Filter

Fortunately the inverse DTFT for an ideal differentiator combined with a lowpass filter can also be analytically evaluated. The ideal amplitude response is the same as [link] and [link] but, since this case has an odd symmetric impulse response, the inverse DTFT uses sine functions which for odd N gives
Equation:

$$\widehat{h}_d(n) = \frac{1}{\pi}\int_0^{\omega_0}\left(\frac{1}{\pi}\omega\right)\sin(\omega n)\,d\omega = \frac{\sin(\omega_0 n)}{\pi^2 n^2} - \frac{\omega_0\cos(\omega_0 n)}{\pi^2 n}$$

with the indeterminate $\widehat{h}_d(0) = 0$. This is now truncated and shifted by $M = (N-1)/2$ to give the optimal, causal length-N FIR filter coefficients as
Equation:

$$h(n) = \frac{\sin(\omega_0(n-M))}{\pi^2(n-M)^2} - \frac{\omega_0\cos(\omega_0(n-M))}{\pi^2(n-M)} \quad \text{for } 0 \le n \le N-1$$

and $h(n) = 0$ otherwise. Again the corresponding derivation for an even length gives the same result as in [link]. Note this very general single formula includes as special cases the odd and even length full band ($\omega_0 = \pi$) differentiator given in [link]. Also note that for a full band differentiator, an even length is

much preferred because of the zero at $\omega = \pi$ for an odd length. However, for the differentiator with a lowpass filter, the zero aids in the lowpass filtering and, therefore, might be an advantage.

Hilbert Transformer

The inverse DTFT for an ideal Hilbert transform [link] combined with a lowpass filter can also be analytically evaluated. The ideal amplitude response is the same as [link] but with a constant phase shift of $\phi = \pi/2$. Since this case has an odd symmetric impulse response, the inverse DTFT uses sine functions which for odd N which gives

Equation:

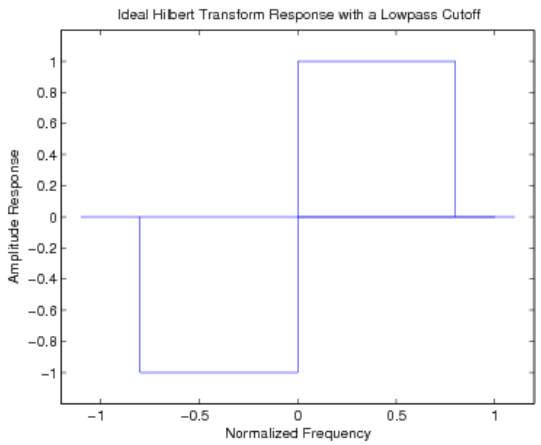
$$\widehat{h}_d\left(n\right) \;=\; \frac{1}{\pi}\int_0^{\omega_0} \sin\left(\omega n\right) d\omega = \frac{1-\cos\left(\omega_0 n\right)}{\pi n}$$

with the indeterminate $\widehat{h}_d\left(0\right) = 0$. This is now truncated and shifted by $M = (N-1)/2$ to give the optimal, causal length-N FIR filter coefficients as

Equation:

$$h\left(n\right) \;=\; \frac{1-\cos\left(\omega_0\left(n-M\right)\right)}{\pi(n-M)} \qquad 0 \leq n \leq N-1$$

and $h(n) = 0$ otherwise. Again the corresponding derivation for an even length gives the same result as in [link]. The ideal amplitude response is shown in [link].



Ideal Frequency Response of an FIR Hilbert Transorm in the Passband and Lowpass Cutoff

Spline Transition Band Design

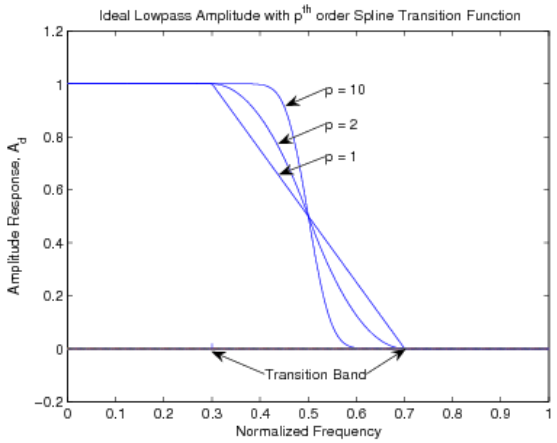
All of the four lowpass filters described above exhibit the Gibbs phenomenon when truncated to a finite length. To remove this effect and to give a more explicit specification of the pass and stopband edges, a transition band is inserted between the pass and stopband. A transition function can be placed in this band to make the total desired amplitude response a continuous function.

If we use a p^{th} order spline as the transition function, the effect of adding this transition band to the basic lowpass filter ideal amplitude given in [link] is to multiply the ideal impulse response in [link] by a the P^{th} power of a sinc function to give

Equation:

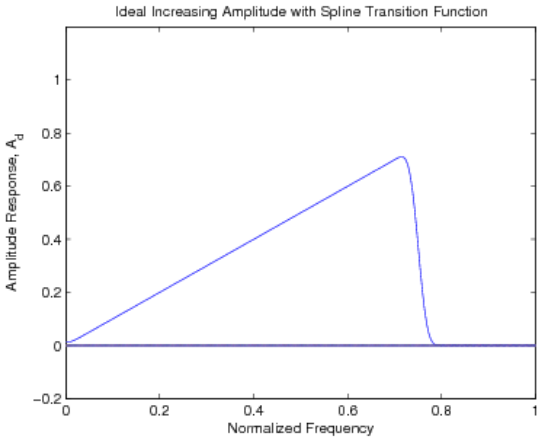
$$\widehat{h}_d(n) = \frac{\sin(\omega_0 n)}{\pi n}\left(\frac{\sin(\Delta n/p)}{\Delta n/p}\right)^p$$

where $\omega_0 = (\omega_s + \omega_p)/2$ is the average band edge and $\Delta = (\omega_s - \omega_p)/2$ is half the transition band width in radians per second normalized for one sample per second sampling rate [link], [link], [link]. The spline produces a transition function which consists of p segments of p^{th} order polynomials connected together so that $p-1$ derivatives are continuous at the junctions.



Ideal Lowpass Filter Amplitude with Order-p Spline Transition Function

The optimal value of the exponent p is chosen as $p = 0.624(f_s - f_p)N$ (for a unity sampling rate) which minimizes the approximation error [link]. Each of the four ideal lowpass filters derived in the previous can have a transition band added simply by multiplying their impulse response by the sinc weighting function as illustrated in [link]. [link] shows an ideal unity gain filter amplitude response with examples of first, second, and tenth order spline transition functions. [link] shows the ideal responses of the linear gain filter with fourth order spline transition function.



Ideal Increasing Amplitude Filter with Spline Transition Function

The Optimal Multiband Least Squared Error Design Method

The optimal multiband design method consists of two somewhat independent parts. The first is the design of an optimal least squares lowpass filter with a transition band as described above or as calculated by an inverse FFT. The second part builds an optimal multiband filter from a combination of these optimal lowpass filters and is the main point of this [link].

The unweighted least squared error linear phase FIR filter design problem is to find the filter coefficients that minimize the error defined by
Equation:

$$q = \int_0^{\pi} |A(\omega) - A_d(\omega)|^2 \, d\omega$$

where $A(\omega)$ is the amplitude frequency response of the actual filter and $A_d(\omega)$ is the desired ideal amplitude response. This is done by truncating the inverse discrete time Fourier transform of $A_d(\omega)$. The difficulty is the analytical evaluation of the integral in the inverse transform [link]. If a spline transition function is used, an analytical formula can be derived for the filter that minimizes [link]. The details of this result can be found in [link], [link].

The infinitely long filters designed from the inverse discrete time Fourier transform of the ideal response have a frequency response which is the same as the ideal and, therefore, has no error. An ideal desired amplitude response can be formulated as the sum of simpler ideal lowpass filters, differentiators or Hilbert transformers together with their spline transition functions by
Equation:

$$A_d(\omega) = \sum_k K_k \, A_{d_k}(\omega).$$

where $A_{d_k}(\omega)$ is the desired lowpass response with a transition band in the k^{th} band such as given in [link] or [link] and the K_k are chosen to build the desired $A_d(\omega)$. These A_{d_k} are the forms considered in the previous section along with any others that have analytical inverse DTFTs such as polynomials. Because of the linearity of the Fourier transform, a multiband ideal response can be constructed by simply adding and subtracting the impulse response of appropriate ideal lowpass filters.
Equation:

$$\widehat{h}_d(n) = \sum_k K_k \,\mathrm{IDTFT}\left\{A_{d_k}(\omega)\right\}$$

Equation:

$$\widehat{h}_d(n) = \sum_k K_k \widehat{h}_{d_k}(n)$$

Because of the orthogonality of the basis functions of the Fourier transform, the truncated sequence of the infinitely long impulse response $\widehat{h}_d(n)$ will give an optimal approximation to $A_d(\omega)$ in a least squares sense. This argument allows no error weighting or “don't care" transition bands or traditional windowing methods. It does, however, allow the optimized spline transition functions [link], [link].

Using these facts, an optimal multiband filter can be built up by successively adding and subtracting the impulse responses of optimal lowpass filters as done in [link]. For example a bandpass filter that approximates zero for $0 < \omega < \omega_1$, has a spline transition band for $\omega_1 < \omega < \omega_2$, approximates one (or some other constant) for $\omega_2 < \omega < \omega_3$, has an independent second transition band for $\omega_3 < \omega < \omega_4$, and finally approximates zero for $\omega_4 < \omega < \pi$ can be designed by first designing a simple lowpass filter with transition band $\omega_3 < \omega < \omega_4$ and then subtracting from its impulse response the impulse of a second lowpass filter designed with a transition band $\omega_1 < \omega < \omega_2$. A filter with two or more passbands can be designed by adding the impulse responses of two or more single passband filters.

Indeed, a completely general design method can be formulated by alternately adding and subtracting lowpass filters starting with the highest frequency transition band and moving sequentially down to the lowest. If the ideal frequency response is not zero at $\omega = \pi$, then one starts with a constant frequency response (an impulse in the time domain) and subtracts a lowpass filter (remember the length must be odd for this case). By scaling each lowpass filter, different gains are obtained in each band.

Care must be taken that the constructed spline transition function properly fit the bands on both sides. This will not automatically happen if there are two adjacent bands with different slopes connected by one transitions function which are simply added together. It will automatically happen if each passband is separated by a stopband or if adjacent bands have the same slopes.

A Matlab Filter Design Program

A Matlab [link] program named fir3.m is given in the appendix of this book that will design optimal filters using the method described in the previous section. This particular program requires constant but arbitrary passband gains and uses a format for specifications similar to remez() in Matlab. It constructs the multiband filter from [link] by adding and subtracting optimal lowpass filters designed from the formula in [link] and calculated in the second program named fir3lp.m .

The main program is given an even length vector f containing the normalized pass and stopband edges, including $f = 0$ and $f = 1$. It is also given an even length vector m containing the ideal response at each frequency in f . Because the lowpass filter has a constant passband, the ideal response of the multiband filter will have constant passbands. This means m will consist of adjacent terms that are equal. An example Matlab function call is given in the next section.

The simple program listed in the appendix will design filters with constant gains in multiple passbands. From its construction it is easy to see how adding the use of the linear gain lowpass filter to the unity gain passband lowpass filter would allow designing optimal filters with linear gains in the passbands. By adding all four basic lowpass designs a calling program could be written that would automatically design one filter with a combination of all four characteristics. If the real and imaginary parts of a desired complex frequency response can be given in terms of the basic filters, nonlinear phase filters can be designed also.

The programs are written to be consistent with Matlab's convention of normalizing for two samples per second sampling rate. The equations most of this book, however, are normalized for one sample per second.

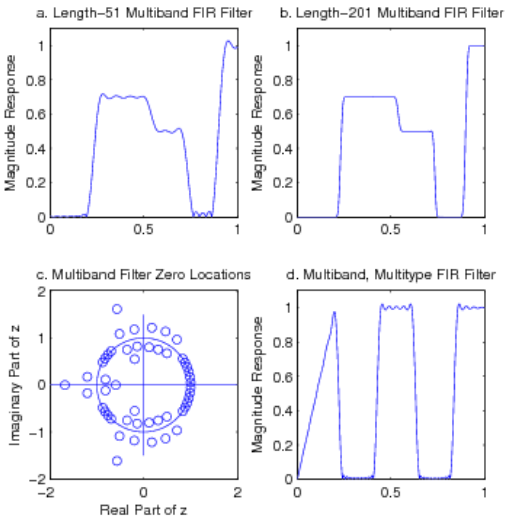
Design Examples

To show the results of using this new design approach, two examples of multiband filter design are presented here. The first is a filter with a stopband from $\omega = 0$ to $\omega = 0.2$, a transition band from $\omega = 0.2$ to $\omega = 0.25$, a passband with gain equal to 0.7 from $\omega = 0.25$ to $\omega = 0.5$, a transition band from $\omega = 0.5$ to $\omega = 0.55$, a passband with gain equal to 0.5 from $\omega = 0.55$ to $\omega = 0.7$, a transition band from $\omega = 0.7$ to $\omega = 0.73$, a stopband from $\omega = 0.73$ to $\omega = 0.85$, a transition band from $\omega = 0.85$ to $\omega = 0.9$, and a passband with gain equal one from $\omega = 0.9$ to $\omega = 1$. This is called with the Matlab program by

```
h  = fir3(51,[0 .2 .25 .5 .55 .7 .73 .85 .9 1],
[0 0 .7 .7.5 .5 0 0 1 1])
```

and the amplitude response plot shown in [link]a. The response for length of $N = 101$ is shown in Figures [link]b and in [link]c the zero locations are given.

As an example of how versatile this approach can be, a length-101 linear phase multiband FIR filter was designed with different types of filtering being done in different bands. The signal with frequencies in the band from $0 < f < 0.2$ is differentiated, in the band from $0.23 < f < 0.4$ is rejected, in $0.43 < f < 0.6$ is Hilbert transformed, in $0.63 < f < 0.8$ is rejected, and $0.83 < f < 1.0$ is highpass filtered. In the transition bands between each of these processing bands, there is an optimal spline transition function. The amplitude response is shown in Figure 7d. This is a truly versatile multiband design technique with the only major limitation being that weighting is not possible. However, that limitation is removed in the next secession.



Frequency Response and Zero Locations of FIR Filters Designed by Least Squared Error

Weighted Least Integral Squares FIR Filter Design

If the FIR filter design problem is posed as a weighted integral squared error approximation problem, a simple analytical design formula as in [link] or [link] is not possible (Recall that it is possible to easily introduce weights in the discrete approximation problem [link]). In this section we consider a multiband generalization [link] of an approach which is a mixture of analytical formulas and numerical solution of Toeplitz plus Hankel matrices which have been presented in [link], [link], [link].

The most general definition of the linear phase weighted least squares FIR filter design problem [link], [link], [link] defines the error measure as in Equation 10 from Constrained Approximation and Mixed Criteria by
Equation:

$$q = \frac{1}{\pi} \int_{\Omega} W(\omega) |A_d(\omega) - A(\omega)|^2 d\omega.$$

where Ω is the set of frequencies that contribute to the error.

We set up the conditions for minimizing the error in [link] for odd N by using the same approach used in [link] which substitutes
Equation:

$$A(\omega) = \sum_{n=0}^{M} a(n) \cos(\omega n)$$

from Equation 34 from FIR Digital Filters into [link], differentiates q in respect to each $a(m)$, and then sets it equal to zero to give
Equation:

$$\frac{1}{\pi}\int_{\Omega} W(\omega)\, A_d(\omega)\, \cos(\omega m)\, d\omega = \sum_{n=0}^{M} a(n)\frac{2}{\pi}\int_{\Omega} W(\omega)\ \cos(\omega n)\ \cos(\omega m)\, d\omega$$

where we can obtain the $h(n)$ from the $a(n)$ by the scaling and shifting in Equation 35 from FIR Digital Filters. We denote this in matrix form by
Equation:

$$\mathbf{A_w} = \mathbf{C_w}\,\mathbf{a}$$

with the elements of the $M+1$ by $M+1$ matrix $\mathbf{C_w}$ as
Equation:

$$c_w(m,n) = \frac{2}{\pi}\int_{\Omega} W(\omega)\ \cos(\omega n)\ \cos(\omega m)\, d\omega$$

and the $M+1$ by 1 vector of intermediate values $\mathbf{a_w}$ is given by an inverse DTFT of the weighted ideal amplitude response in
Equation:

$$A_w(n) = \frac{1}{\pi}\int_{\Omega} W(\omega)\, A_d(\omega)\ \cos(\omega n)\, d\omega \qquad \text{for } 1 \le n \le N-1$$

and
Equation:

$$A_w(0) = \frac{1}{2\pi}\int_{\Omega} W(\omega)\, A_d(\omega)\, d\omega.$$

Solving [link] for the optimal $\mathbf{a}$ which minimizes the integral weighted squared error [link] is formally done by
Equation:

$$\mathbf{a} = \mathbf{C_w^{-1}}\,\mathbf{A_w}$$

and more accurately done by special numerical algorithms. The case for even N is easily derived by using Equation 40 from FIR Digital Filters to derive [link]. The actual length-N filter coefficients $h(n)$ are then found from $a(n)$ using Equation 41 from FIR Digital Filters. Note that if the weighting is unity across the pass, transition, and stop bands, $\mathbf{C_w}$ is the identity matrix. $\mathbf{C_w}$ gives the effects of the weighting.

A similar formula was derived by Fleischer [link], Tufts, Rorabacher and Mosier [link], by Schüssler [link], by Oetken, Parks, and Schüssler [link], and by Burrus, Soewito, and Gopinath [link], [link] in addressing similar problems.

If the integrals in [link] and [link] can be analytically evaluated, the solution of the weighted squared error approximation is obtained by solving $M+1$ equations. Fortunately these equations can be analytically evaluated for several interesting cases as was done for the unweighted case. The even length-N case is derived in a similar way using Equation 40 from FIR Digital Filters and Equation 41 from FIR Digital Filters. An alternate formulation could modify the first column.

In most practical situations where specifications are set in the frequency domain, these filters are described in terms of frequency bands. We have already seen the idea of single pass, stop, and transition bands. We now allow multiple pass and stopbands separated by multiple transition bands. In order to obtain analytical solutions of [link] and [link] and to be consistent with usual practice, we restrict ourselves to constant weights over each separately defined frequency band. The error in [link] now becomes

Equation:

$$q \;=\; \frac{1}{\pi}\sum_{k}\left[W_k\int_{\omega_k}^{\omega_{k+1}}\left|A_{d_k}\left(\omega\right)-A\left(\omega\right)\right|^2 d\omega\right].$$

where the weights are constant over each band and are given by $W\left(\omega\right)=W_k$ in the k^{th} band defined by $\omega_k<\omega<\omega_{k+1}$. The desired amplitude $A_{d_k}\left(\omega\right)$ is likewise defined in the k^{th} band and is hopefully simple enough to allow analytical evaluation of the formula [link] for the ideal impulse response.

The form of [link] causes [link] to become

Equation:

$$c_w\left(m,n\right) \;=\; \frac{2}{\pi}\sum_{k}\left[W_k\int_{\omega_k}^{\omega_{k+1}}\cos\left(\omega n\right)\;\cos\left(\omega m\right)d\omega\right].$$

which has an analytical solution as given by

Equation:

$$c_w\left(m,n\right) \;=\; \frac{1}{\pi}\sum_{k}W_k\left[\frac{\sin\left(n-m\right)\omega_{k+1}-\sin\left(n-m\right)\omega_k}{\left(n-m\right)}+\frac{\sin\left(n+m\right)\omega_{k+1}-\sin\left(n+m\right)\omega_k}{\left(n+m\right)}\right]$$

which for F band edges has terms that are indeterminate for $n=m\neq 0$ with values

Equation:

$$c_w\left(n,n\right) \;=\; \frac{1}{\pi}\left\{\sum_{k=1}^{F-2}W_k\left[\left(\omega_{k+1}-\omega_k\right)+\frac{\sin\left(2n\omega_{k+1}\right)-\sin\left(2n\omega_k\right)}{2n}\right]+W_{F-1}\pi\right\}$$

and for $n=m=0$ as

Equation:

$$c_w\left(0,0\right) \;=\; \frac{1}{\pi}\left\{\sum_{k=1}^{F-2}W_k 2\left(\omega_{k+1}-\omega_k\right)+W_{F-1}2\pi\right\}$$

Since the matrix elements are functions of $(n-m)$ and $(n+m)$, $\mathbf{C}$ is the sum of a Toeplitz and a Hankel matrix. This matrix can always be calculated and it simply depends on the set of band edges ω_k and the band weights W_k but not on the ideal amplitude response $A_d(\omega)$. The case for even N is similar but uses Equation 8 from Constrained Approximation and Mixed Criteria rather than Equation 7 from Constrained Approximation and Mixed Criteria with [link] to derive an appropriate form of [link].

If there are F distinct band edges ω_k, the first and last are $\omega_1 = 0$ and $\omega_F = \pi$. This means part of the first term in the sum of [link] is always zero and part of the last is zero except when $n = m = 0$ where it is π. Using these facts allows [link] to be written
Equation:

$$c_w(m,n) = \frac{1}{\pi}\sum_{k=1}^{F-2}(W_k - W_{k+1})\left[\frac{\sin(n-m)\omega_{k+1}}{n-m} + \frac{\sin(n+m)\omega_{k+1}}{n+m}\right]$$

which, together with appropriately modified [link] and [link], are good forms for programming. The Matlab program in the appendix contains the details.

Applying the form of [link] to [link] gives
Equation:

$$a_w(n) = \frac{1}{\pi}\sum_k\left[W_k\int_{\omega_k}^{\omega_{k+1}} A_{d_k}(\omega)\ \cos(\omega n)\, d\omega\right].$$

These integrals have been evaluated for the four basic filter types - constant gain passband lowpass filter, linear gain passband lowpass filter, differentiator plus lowpass filter, and Hilbert transformer plus lowpass filter - giving simple design formulas in [link], [link], [link], and [link].

Each basic filter type plus the effects of a transition band can be calculated and combined according to [link]. An example low pass filter with a weight of W_1 in the passband and W_2 in the transition band is given for odd N gives for the intermediate coefficients $\widehat{h}_w(n)$ from [link] are
Equation:

$$a_w(n) = W_1\left[\frac{\sin(\omega_2 n) - \sin(\omega_1 n)}{\pi n}\right] + W_2\left[\frac{\sin(\omega_0 n)}{\pi n}\left(\frac{\sin(\Delta n/p)}{\Delta n/p}\right)^p - \frac{\sin(\omega_1 n)}{\pi n}\right]$$

A similar expression can be derived for even N using Equation 8 from Constrained Approximation and Mixed Criteria.

This means the left hand vector in [link] can be calculated as a weighted sum of inverse DTFTs such as in [link] if the ideal desired amplitude can be constructed from the four basic types in FIR Digital Filters, each with optimal transition bands.

If one or more of the integrals in [link] has no analytical solution, $a_w(n)$ can be calculated numerically using a truncated weighted sum of inverse DFTs of a dense sampling of $A_{d_k}(\omega)$ or made up of the passbands calculated from inverse DFTs and the transition bands added by multiplying appropriately by sinc functions since constructing an optimal spline transition function to be sampled would not be easy.

This gives a very powerful design method that allows multi band weighted least squares design of FIR filters. The calculation of the matrix $\mathbf{C_w}$ in [link] is always possible using [link]. Because using a true "don't care" transition band with a weight of zero might causes ill conditioning of [link] for $(f_s - f_p)N > 12$ as discussed in [link], one can add a spline transition function in $A_d(\omega)$ to the definition in [link] as done in [link] and [link]. A very small weight used in the transition bands together with a spline transition function will improve the conditioning of [link] with minor degradation of the optimality. This point needs further evaluation.

By using an inverse FFT perhaps plus a sinc induced transition function to calculate the components of [link], this method can be used to design arbitrary shaped passbands. It can also be used for complex approximation by applying it to the real and imaginary parts of the desired $H_d(\omega)$ separately and using the full, nonsymmetric $h(n)$.

The form of the simultaneous equations [link] that must be solved to design a filter by this method is interesting. If the weights in all pass, stop, and transition bands are unity, the $\mathbf{C_w}$ matrix is the identity matrix and $\widehat{\mathbf{h}}_\mathbf{w}$ contains the filter coefficients. As the weights become less and less uniform or equal, the $\mathbf{C_w}$ matrix becomes poorer conditioned. If the weights for the transition bands are zero, it is the smallest eigenvalues of $\mathbf{C_w}$ that control the actual amplitude response $A(\omega)$ in the transition bands. This explains why numerical errors in solving [link] show up primarily in the transition bands. It also suggests this effect can be reduced by allowing a small weight in the transition bands. Indeed, one can design long filters by using spline transition functions with a small weight which then allows different pass and stopband weights.

Matlab Programs

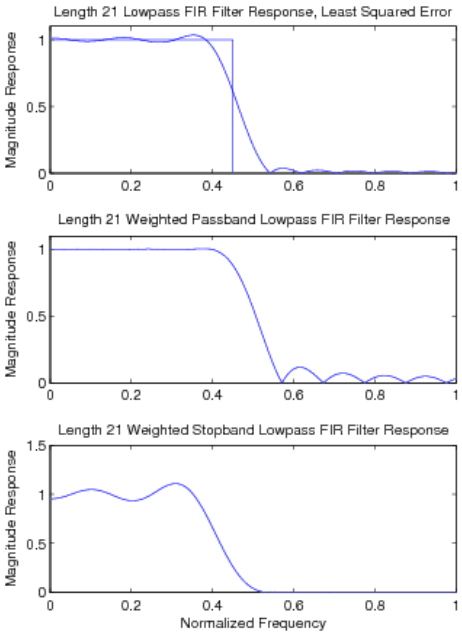
A Matlab program for designing multiband FIR filters using the weighted least squares approximation described in [link] and [link] above is included in the Appendix. The program assumes passbands and stopbands that alternate with transition bands. The passbands are assumed to have constant gain for simplicity but that could be generalized with the results from FIR Digital Filters. The first for loop constructs the $\widehat{h}_w(n)$ in [link] by sequentially designing weighted bandpass filters and a separately weighted transition band similar to the example in [link]. These are added together in this loop to give the vector h_w in [link]. The second for loop constructs the $\mathbf{C}$ matrix in [link] using the formula [link]. Care must be taken to correctly calculate the indeterminate values of sinc(0) and to properly include the effects of $\omega_F = \pi$.

When one uses zero weights in the transition bands, the $\mathbf{C}$ matrix becomes ill conditioned when the product of the filter length N and the sum of the transition band widths in Hertz is much above 12. This is an approximate rule which is somewhat affected by different passband and stopband weights, but it gives an indication of when numerical problems will occur. To reduce this problem, the program includes optimal spline transition functions so that a small weight can be used to improve the conditioning of $\mathbf{C}$ with minimal effect on the optimality of the approximation.

Examples

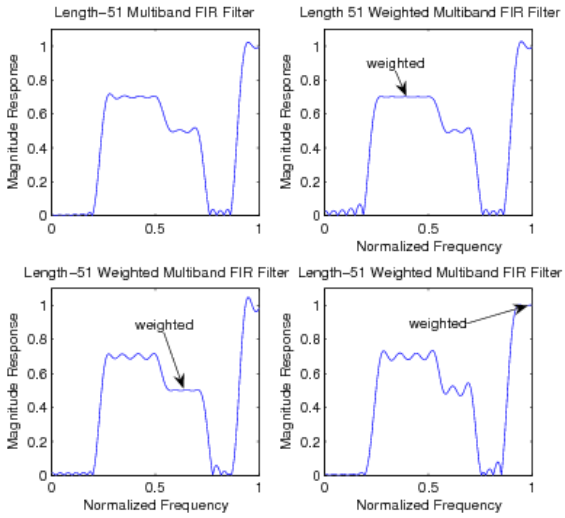
To illustrate the effects of using a weighted least squared error design criterion, a simple length-21 linear phase lowpass FIR filter was designed, with unity weighting in the pass and stop bands and zero weighting in the transition band. The frequency response is shown in [link]a. The same filter is designed with a weight of 100 in the passband and the response is shown in [link]b and the case for a weight of

100 in the stopband is shown in [link]c. It is instructive to design many example filters and observe the effects of different weights, use of spline vs zero weight transition bands, and the effects of the transition band width on the pass and stopband performance.



Frequency Response of Length-21 FIR Filter Designed by Weighted Least Squared Error

The same specifications that were used in the design using optimal spline transition functions of [link]a is used in the design here with unity weights in the stop and passbands and zero weights in the transition bands. The result is fairly similar to the spline function design and is shown in [link]a for $N = 51$. For lengths above around 131, numerical errors resulted in erratic performance in the transition bands. Indeed for this case, the use of the spline method would probably be superior. The advantage of the weighted least squares method is illustrated in [link]b where the same specifications are used but with a weight of 100 in the first passband, in [link]c where a weight of 100 is used in the second passband, and in [link]d where a weight of 100 is used in the third passband. This use of weights is impossible using the spline method or any windowing method.



Frequency Response of Length-15 FIR Filter Designed by Least Squared Error

Section Conclusions

This section has derived the four basic ideal lowpass filters: the constant gain passband lowpass filter, the linearly increasing gain passband lowpass filter, the differentiator with a lowpass filter, and the Hilbert transformer with a lowpass filter. It is shown that each of these can be modified to allow a spline transition function by a simple weighting function.

Because of using an L_2 approximation error criterion and because of the orthogonality of the basis functions of the Fourier transform, it is shown that an optimal multiband filter can be built from the linear combination of these optimal building blocks. This new filter design method has the flexibility of the Parks-McClellan algorithm but the simplicity of the windowing methods. It is extremely fast and has no numerical problems. Unlike the windowing methods, the new method allows explicit independent control of multiple transition band edges and gives an optimal design. Its only limitation is not allowing error weighting.

We then derived a second method that likewise allowed multiple pass, stop, and transition bands with arbitrary band edges, but also allowed independent weighting of each frequency band. There are two limitations on this method. For long filters with wide transition bands with zero weights and where $N(f_p - f_s) > 12$, the equations that must be solved are ill conditioned. This can be partially addressed using optimal spline functions with small weights in the transition bands. The second problem is that solving a large number of simultaneous equations can be slow and require considerable memory. These problems might be addressed by using special Toeplitz or Toeplitz plus Hankel algorithms [link] or some iterative method.

When should these methods be used? The second method which minimizes the weighted integral squared error should be used anytime the original problem dictates a squared error criterion and the product of the length and transition band width is less than twelve, $N\left(f_p - f_s\right) < 12$. These conditions are often met because the squared error is a measure of the signal or noise energy and one seldom wants a long filter

and a wide transition band. Even though this method requires solution of a set of simultaneous equations and is, therefore, slower than the spline transition function method, it executes in a few seconds on a PC or workstation and allows independent weighting of different frequency bands.

The first method which uses spline functions in the ideal response transition bands will design essentially arbitrarily long filters very quickly but it will not allow any error weighting. Although artificial transition functions are used in the ideal response, the optimized spline functions are very close to the response actually obtained by the second method with zero weighting in the transition band. This means the optimal approximation to the ideal response with spline functions transition bands is close to that obtained using the second numerical method. Comparisons of these effects for a single band can be found in [link]. If a Chebyshev approximation is desired, the Parks-McClellan method should be used although it too has numerical problems for long filters with wide transition bands. If different error measures are wanted in different bands, the iterative reweighted least squares (IRLS) algorithm [link] should be used. Recent research suggest that for many practical signal specifications, a mixture of Chebyshev and least squares is appropriate with no explicit transition bands [link].

If the equations that must be solved to obtain the optimal filter coefficients are ill-conditioned, an orthogonalization procedure can be used to improve the conditioning [link].

Characteristics of Optimal Filters

Gibbs phenomenon, transition band, pole-zero plots, etc.

Complex and Minimum Phase Approximation

Here we talk about which methods also solve the complex approximation problem. We talk about the minimum phase filter.

Chebyshev or Equal Ripple Error Approximation Filters

If one poses the FIR filter design problem by requiring the maximum error over certain bands of frequencies be minimized, we call the resulting filter a Chebyshev filter or an equal ripple filter. The fact that the minimization of the Chebyshev or L_∞ error results in an equal ripple error comes from the alternation theorem. This very powerful theorem allows one to minimize the Chebyshev error by directly constructing an equal ripple approximation with the proper number of ripples. That is the basis of several very effective algorithms, including the Remez exchange algorithm.

There are several ways one could pose the Chebyshev FIR filter design problem. For a simple length-N linear phase, lowpass filter with a transition band, if one considers the length N, the passband ripple δ_p, the stopband ripple δ_s, and the transition bandwidth $\Delta = \omega_s - \omega_p$, then one can fix or constrain any three of them and minimize the fourth. Or, as Parks and McClellan do, fix the band edges, ω_p and ω_s, and the ratio of δ_p and δ_s and minimize one of them.

The Chebyshev error measure is often used for approximation in digital filter design. This is particularly true when the signals and/or noise are narrow band or single frequency or when one wants to minimize worst case possibilities. Theoretical justification for its use has been given by Weisburn, Parks, and Shenoy [link]. For FIR filter design, the Parks-McClellan formulation of the filter design problem and application of the Remez exchange algorithm is most commonly used [link], [link]. It is a particularly interesting and powerful method that should be studied and understood to be fully utilized.

Linear programming was used earlier [link], [link], [link] but dropped out of favor when the Parks-McClellan algorithm was introduced. It is now becoming more popular again because of more powerful computers, better algorithms [link], [link], and linear programming's ability to allow a variety of constraints [link].

Still another approach to achieving a Chebyshev approximation is to minimize the p^{th} power of the error using a large value of p or to use an iterative scheme that solves a weighted least squared error with the weights at each stage determined by the error of the previous stage [link]. Still another design method that produces an equal ripple error approximation uses a constrained least squared error criterion [link], [link] which results in a Chebyshev solution if tight constraints are imposed.

The early work by Herrmann and Schüssler [link], [link] and the algorithm by Hofstetter, Oppenheim, and Siegel [link], [link] posed and solved a similar problem but they had only approximate control of ω_o (or ω_p or ω_s) and always achieved the “extra ripple" design. Given the proper specifications, the Parks-McClellan algorithm could design any filter that the Hofstetter-Oppenheim-Siegel algorithm could, but the opposite is not true. This seems to be one of the reasons the Hofstetter-Oppenheim-Siegel algorithm is not commonly used.

The Linear Phase FIR Filter Chebyshev Approximation Problem

The Chebyshev error is defined as the maximum difference between the actual and desired response over a band or several bands of frequencies. This is
Equation:

$$\varepsilon = \max_{\omega \in \Omega} \left| A(\omega) - A_d(\omega) \right|$$

where Ω is the union of the bands of frequencies that the approximation is over [link], [link]. The approximation problem in filter design is to choose the filter coefficients to minimize ε.

One way to minimize ε is to set up the frequency response in four equations for the four types of linear phase FIR filters as done in Equation 34 from FIR Digital Filters, Equation 40 from FIR Digital Filters, and the corresponding sine expressions. An alternative approach [link] uses the fact that all four can be obtained from the odd-length, even-symmetry type 1 and uses only Equation 34 from FIR Digital Filters. From one of these frequency response representations together with powerful **Alternation Theorem** several optimization schemes can be developed.

If the amplitude response for odd L is expressed as a sum of R cosine terms
Equation:

$$A\left(\omega\right) \ = \ \sum_{n=0}^{R-1} a\left(n\right) \ \cos\left(\omega n\right)$$

or for even L
Equation:

$$A\left(\omega\right) \ = \ \sum_{n=1}^{R} a\left(n\right) \ \cos\left(\omega\left(n-1/2\right)\right)$$

with $R = M + 1 = \frac{L+1}{2}$ for odd length-L and $R = L/2$ for even length-L, as derived in Equation 34 from FIR Digital Filters and Equation 40 FIR Digital Filters, then

Theorem 1
If $A(\omega)$ is the linear combination of R cosine functions given in [link] or [link], the necessary and sufficient conditions for $A(\omega)$ to be the least Chebyshev error approximation to $A_d\left(\omega\right)$ over $\omega \in \Omega$ are: The error function, $\epsilon\left(\omega\right) = A\left(\omega\right) - A_d\left(\omega\right)$ have **at least**$R + 1$ extremal frequencies in Ω. The extremal frequencies are ordered points $\omega_1 < \omega_2 < \cdots < \omega_{R+1}$ such that $\epsilon\left(\omega_k\right) = -\epsilon\left(\omega_{k+1}\right)$ and $\max_{\omega \in \Omega} \left|\epsilon\left(\omega\right)\right| = \left|\epsilon\left(\omega_k\right)\right|$ for $k = 1, 2, \cdots, R + 1$.

The alternation theorem [link], [link] states that the minimum Chebyshev error has at least $R + 1$ extremal frequencies. This is stated mathematically by
Equation:

$$A\left(\omega_k\right) \ = \ A_d\left(\omega_k\right) + \left(-1\right)^k \delta$$

for $k = 0, 1, 2, \cdots, R$, where the ω_k are the ordered extremal frequencies where the equal ripple error has maximum value. In other words, the optimal solution to the linear phase FIR filter design problem will have an equal ripple error function with a required number of ripples. How all of these characteristics relate can be rather complicated and good designs require experience [link]. When applied to other approximation problems, care must be taken to ensure the approximating functions satisfy the “Haar conditions" or other restrictions [link], [link], [link], [link].

Chebyshev Approximation by Linear Programming

It is possible to pose the Chebyshev approximation problem in filter design as a linear programming optimization problem [link], [link], [link], [link]. The error definition in [link] can be written as an inequality by
Equation:

$$A_d(\omega) - \delta \leq A(\omega) \leq A_d(\omega) + \delta$$

where the scalar δ is minimized.

The inequalities in [link] can be written as
Equation:

$$A \leq A_d + \delta$$

Equation:

$$-A \leq -A_d + \delta$$

or
Equation:

$$A - \delta \leq A_d$$

Equation:

$$-A - \delta \leq -A_d$$

which can be combined into one matrix inequality using Equation 48 from FIR Digital Filters by
Equation:

$$\begin{bmatrix} C & -1 \\ -C & -1 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix} \leq \begin{bmatrix} A_d \\ -A_d \end{bmatrix}.$$

If δ is minimized, the optimal Chebyshev approximation is achieved. This is done by minimizing
Equation:

$$\varepsilon = \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a \\ \delta \end{bmatrix}$$

which, together with the inequality of [link], is in the form of the dual problem in linear programming [link], [link], [link].

This can be solved using the lp() command from the Optimization Toolbox with Matlab [link], the Meteor software system [link], CPlex [link], or Karmarkar's algorithm [link], [link]. The Matlab lp command is implemented in an m-file using a form of quadratic programming algorithm that is not well suited to our filter design problem. Meteor is a linear programming system using the simplex algorithm written in Pascal by Ken Steiglitz especially for filter design. It has been compiled on a variety of computers and efficiently designs filters over 100 in length. The Karmarkar program written by Lang is

a relatively short m-file that is not particularly fast but is robust and can design filters on the order of length-100. CPlex is a proprietary program that can be used alone or called from Fortran programs and is particularly robust and fast.

A Matlab program that applies its linear programming function lp.m to [link],[link] for linear phase FIR filter design is given by:

```
% lpdesign.m  Design an FIR filter from L, f1, f2, and LF using LP.
%  L is filter length, f1 and f2 are pass and stopband edges, LF is
%  the number of freq samples.  L is odd.  Uses lp.m
%         csb 5/22/91
L1 = fix(LF*f1/(.5-f2+f1));  L2 = LF - L1;     %No. freq samples in
PB, SB
Ad = [ones(L1,1); zeros(L2,1)];                %Samples of ideal
response
f  = [[0:L1-1]*f1/(L1-1), ([0:L2-1]*(.5-f2)/(L2-1) + f2)]';  %Freq
samples
M  = (L-1)/2;
C  = cos(2*pi*(f*[0:M]));                      %Freq response matrix
CC = [C, -ones(LF,1); -C, -ones(LF,1)];        %LP matrix
AD = [Ad; -Ad];
c  = [zeros(M+1,1);1];                         %Cost function
x0 = [zeros(M+1,1);max(AD)+1];                 %Starting values
x  = lp(c,CC,AD,[],[],x0);                     %Call the LP
d  = x(M+2);                                   %delta or deviation
a  = x(1:M+1);                                 %Half impulse resp.
h  = [a(M+1:-1:2);2*a(1);a(2:M+1)]./2;         %Impulse response
```

This program has numerical problems for filters longer than 10 or 20 and is fairly slow. The `lp()` function uses an algorithm that seems not well suited to the equations required by filter design. It would be nice to have Meteor written in Matlab, both to show how the Simplex algorithm works, and to have an efficient LP filter design system in Matlab. The above program has been tested using Karmarkar's algorithm [link], [link], [link] as implemented in Matlab by Lang [link]. It proved to be robust and reliable for lengths up to 100 or more. It was faster than the Matlab function but slower than Meteor or CPlex. Its use should be further investigated.

Direct use of quadratic programming and other optimization algorithms seem promising [link], [link], [link], [link], [link], [link], [link], [link], [link], [link]

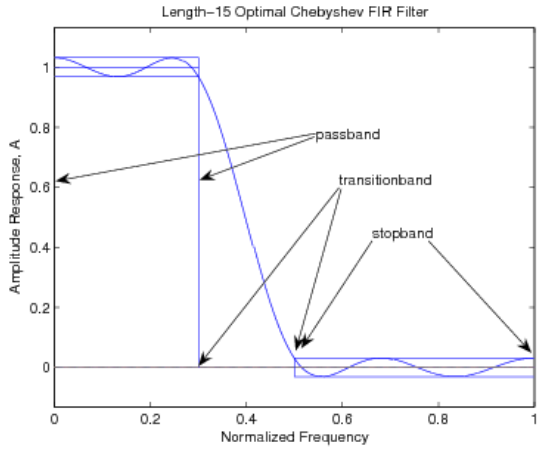
Chebyshev Approximations using the Exchange Algorithms

A very efficient algorithm which uses the results of the alternation theorem is called the **Remez exchange algorithm**. Remez [link], [link], [link] showed that, under rather general conditions, an algorithm that takes a starting estimate of the location of the extremal frequencies and exchanges them with a new set calculated at each iteration will converge to the optimal Chebyshev approximation. The efficiency of this algorithm comes from finding the optimal solution by directly constructing a function that satisfies the alternation theorem rather than minimizing the Chebyshev error as done by the linear programming technique. The Remez exchange algorithm has proven to be well suited to the design of linear phase FIR filters [link], [link], [link].

A particularly useful FIR filter design implementation of the Remez exchange is called the Parks-McClellan algorithm and is described in [link], [link], [link], [link]. It has been implemented in Fortran in [link], [link], [link], [link] and in Matlab in a program at the end of this material. The Matlab program is particularly helpful in understanding how the algorithm works, however, because it does not use any special tricks, it is limited to lengths of 60 or so. Extensions and details can be found in [link], [link], [link], [link], [link], [link], [link], [link], [link], [link], [link]. This is a robust, efficient algorithm that significantly changed DSP when Parks and McClellan first described it in 1972 and has undergone important improvements. Examples are illustrated in [link], [link].

The Basic Parks-McClellan Formulation and Algorithm

Parks and McClellan formulated the basic Chebyshev FIR filter design problem by specifying the desired amplitude response $A(\omega)$ and the transition band edges, then minimizing the weighted Chebyshev error over the pass and stop bands. For the basic lowpass filter illustrated in [link], the pass band edge ω_p and the stop band edge ω_s are specified, the maximum passband error is related to the maximum stop band error by $\delta_p = K\,\delta_s$ and they are minimized.



Amplitude Response of a Length-15 Optimal Chebyshev Filter

Notice that if there is no transition band, i.e. $\omega_p = \omega_s$, that $\delta_p + \delta_s = 1$ and no minimization is possible. While not the case for a least squares approximation, a transition band is necessary for the Chebyshev approximation problem to be well-posed. The effects of a small transition band are large pass and stopband ripple as illustrated in [link]b.

The alternation theorem states that the optimal approximation for this problem will have an error function with $R + 1$ extremal points with alternating signs. The theorem also states that there exists $R + 1$ frequencies such that, if the Chebyshev error at those frequencies are equal and alternate in sign, it will be minimized over the pass band and stop band. Note that there are nine extremal points in the

length-15 example shown in [link], counting those at the band edges in addition to those that are interior to the pass and stopbands. For this case, $R = (L+1)/2$ which agree with the example.

Parks and McClellan applied the Remez exchange algorithm [link] to this filter design problem by writing $R+1$ equations using Equation 37 from FIR Digital Filters and Equation 1 from Design of IIR Filters by Frequency Transformations evaluated at the $R+1$ extremal frequencies with R unknown cosine parameters $a(n)$ and the unknown ripple value, δ. In matrix form this becomes
Equation:

$$\begin{matrix} A_d(\omega_0) \\ A_d(\omega_1) \\ A_d(\omega_2) \\ A_d(\omega_3) \\ \vdots \\ A_d(\omega_R) \end{matrix} = \begin{matrix} \cos(\omega_0 0) & \cos(\omega_0 1) & \cdots & \cos(\omega_0 (R-1)) & 1 \\ \cos(\omega_1 0) & \cos(\omega_1 1) & \cdots & \cos(\omega_1 (R-1)) & -1 \\ \cos(\omega_2 0) & \cos(\omega_2 1) & \cdots & \cos(\omega_2 (R-1)) & 1 \\ \cos(\omega_3 0) & \cos(\omega_3 1) & \cdots & \cos(\omega_3 (R-1)) & -1 \\ \vdots & & & & \vdots \\ \cos(\omega_R 0) & \cos(\omega_R 1) & \cdots & \cos(\omega_R M) & \pm 1 \end{matrix} \begin{matrix} a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(R-1) \\ \delta \end{matrix} .$$

These equations are solved for $a(n)$ and δ using an initial guess as to the location of the extremal frequencies ω_i. This design is optimal but only over the guessed frequencies, and we want optimality over all of the pass and stopbands. Therefore, the amplitude response of the filter is calculated over a dense set of frequency samples using Equation 34 from FIR Digital Filters and a new set of estimates of the extremal frequencies is found from the local minima and maxima and these are used to replace the initial guesses (they are exchanged). This process is iteratively performed until the guaranteed convergence is achieved and the optimal filter is designed.

The detailed steps of the Parks-McClellan algorithm are:

1. Specify the ideal amplitude, $A_d(\omega)$, the stop and pass band edges, ω_p and ω_s, the error weight K where $\delta_p = K\,\delta_s$, and the length L.
2. Choose $R+1$ initial guesses for the extremal frequencies, ω_i, in the bands of approximation, Ω. This is often done uniformly over the pass and stop bands, including $\omega = 0, \omega_p, \omega_s$, and π.
3. Calculate the cosine matrix at the current ω_i and solve [link] for $a(n)$ and δ which are optimal over these current extremal frequencies, ω_i.
4. Using the $a(n)$ or the equivalent $h(n)$ from step 3, evaluate $A(\omega)$ over a dense set of frequencies. This amplitude response will interpolate $A(\omega_i) = A_d(\omega_i) \pm \delta$ at the extremal frequencies.
5. Find $R+1$ new extremal frequencies where the error has a local maximum or minimum and has alternating sign. This includes the band edges.
6. If the largest error is the same as δ found in step 3, then convergence has occured and the optimal filter has been designed, otherwise, exchange the old extremal frequencies ω_i used in step 2 and return to step 3 for the next iteration.
7. This iterative algorithm is guaranteed to converge to the unique optimal solution using almost any starting points in step 2.

This iterative procedure is called a multiple exchange algorithm because all of the extremal frequencies are up-dated each iteration. If only the frequency of the largest error is up-dated each iteration, it is called a single exchange algorithm which also converges but much more slowly. Some modification of the Parks-McClellan method or the Remez exchange algorithm will not converge as a multiple exchange, but will as a single exchange.

The Alternation theorem states that there will be a minimum of $R+1$ extremal frequencies, even for multiband designs with arbitrary $A_d(\omega)$. If $A_d(\omega)$ is piece-wise constant with T transition bands, one can derive the maximum possible number of extremal frequencies and it is $R+2T$. This comes from the maximum number of maxima and minima that a function of the form [link] or [link] can have plus two at the edges of each transition band. For a simple lowpass filter with one passband, one transition band, and one stopband, there will be a minimum of $R+1$ extremal frequencies and a maximum of $R+2$. For a bandpass filter, the maximum is $R+4$. If a design has more than the minimum number of extremal frequencies, it is called an **extra ripple** design. If it has the maximum number, it is called a **maximum ripple** design.

It is interesting to note that at each iteration, the approximation is optimal over that set of extremal frequencies and δ increased over the previous iteration. At convergence, δ has increased to the maximum error over Ω and that is the minimum Chebyshev error.

At each iteration, the exchange of a proper set of extremal frequencies with alternating signs of the errors is always possible. One can show there will never be too few and if there are too many, one uses those corresponding to the largest errors.

In step 4 it is suggested that the amplitude response $A(\omega)$ be calculated over a dense grid in the pass and stopbands and in step 5 the local extremes are found by searching over this dense grid. There are more accurate methods that use bisection methods and/or Newton's method to find the extremal points.

In step 2 it is suggested that the simultaneous equation of [link] be solved. Parks and McClellan [link] use a more efficient and numerically robust method of evaluating δ using a form of Cramer's rule. With that δ, an interpolation method can be used to find $a(n)$. This is faster and allows longer filters to be designed than with the linear algebra based approach described here.

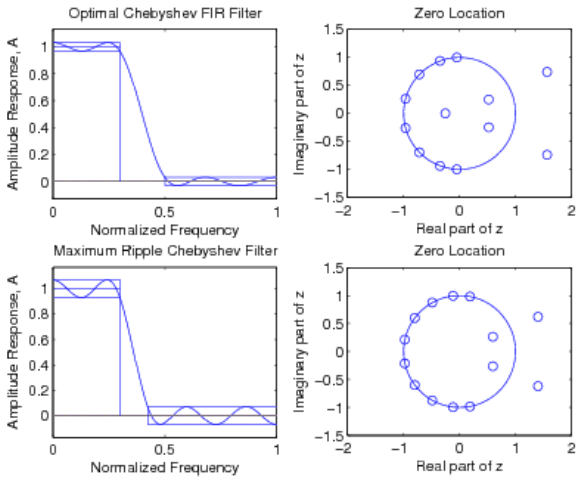
For the low pass filter, this formulation always has an extremal frequency at both pass and stop band edges, ω_p and ω_s, and at $\omega=0$ and/or at $\omega=\pi$. The extra ripple filter has $R+2$ extremal frequencies including both zero and pi. If this algorithm is started with an incorrect number of extremal frequencies in the stop or pass band, the iterations will correct this. It is interesting and informative to plot the frequency response of the filters designed at each iteration of this algorithm and observe how the correction takes place.

The Parks-McClellan algorithm starts with fixed pass and stop band edges then minimizes a weighted form of the pass and stop band error ripple. In some cases it may be more appropriate to fix one of the ripples and minimize the other or to fix both ripples and minimize the transition band width. Indeed Schüssler, Hofstetter, Tufts, and others [link], [link], [link], [link] formulated some of these ideas before Parks and McClellan developed their algorithm. The DSP group at Rice has developed some modifications to these methods and they are presented below.

Examples of the Parks-McClellan Algorithm

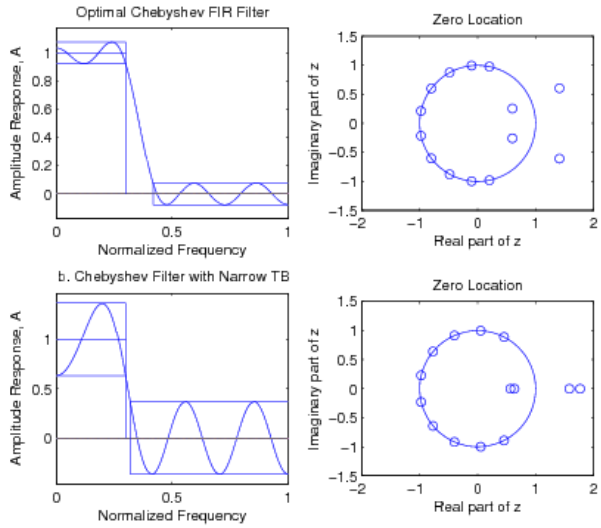
Here we look at several examples of filters designed by the Parks-McClellan algorithm. The examples here are length-15 with that shown in [link]a having a passband $0<f<0.3$, a transition band $0.3<f<0.5$, and a stopband $0.5<f<1$. The number of cosine terms in the frequency response formula is $R=8$, therefore, the alternation theorem says we must have at least $R+1$ extremal points. There are four in the passband, counting the one at zero frequency, the minimum, the maximum, and the minimum at the bandedge. There are five in the stopband, counting the ones at the bandedge and at

$f = 1$. So, the number is nine which is at least $R + 1$. However, in [link]c, there are ten extremal points but that is also at least 9, so it also is optimal. For a low pass filter, the maximum number of extremal points is $R + 2$ and that is what this filter has. This special case is called the “maximum ripple" case.



Amplitude Response of Length-15 Optimal Chebyshev Filters

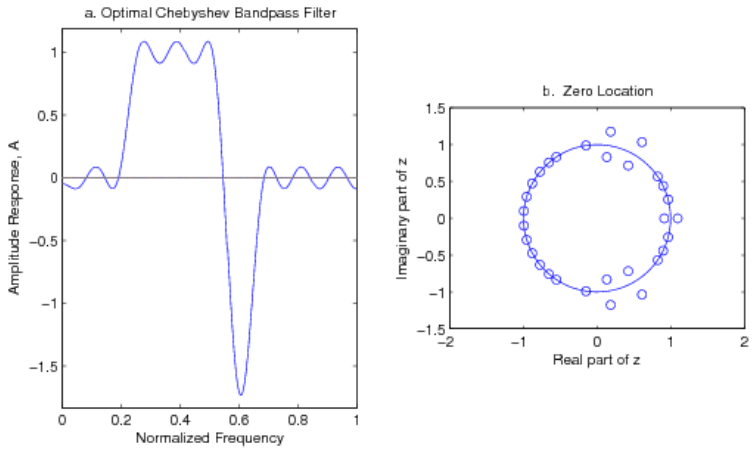
It is possible to have ripples that do not touch the maximum value and, therefore, are not considered extremal points. That is illustrated in [link]a. The effects of a narrow transitionband are illustrated in [link]c. Note the zero locations for these filters and how they relate to the amplitude response.



Amplitude Response of Length-15 Optimal

Chebyshev Filters

To illustrate some of the unexpected behavior that optimal filter designs can have, consider the bandpass filter amplitude response shown in [link]. Here we have a length-31 Chebyshev bandpass filter with a stopband $0 < f < 0.2$, a transition band $0.2 < f < 0.25$, a passband $0.25 < f < 0.5$, another transitionband $0.5 < f < 0.68$, and a stopband $0.68 < f < 1$. The asymmetric transition bands cause large response in the transition band around $f = 0.6$. However, this filter is optimal since the deviation occurs in part of the frequency band that is not included in the optimization criterion. If you think you don't care what happens in the transition bands, you may change your mind with this kind of behavior.



Amplitude Response of Length-31 Optimal Chebyshev Bandpass Filter

The Modified Parks-McClellan Algorithm

If one wants to fix the pass band ripple and minimize the stop band ripple [link], equation [link] is changed so that the pass band ripple is added to the appropriate top part of the vector A_d of the desired response and the unknown stop band is kept in the lower part of the last column of the cosine matrix C.
Equation:

$$\begin{array}{ccccc}
A_d(\omega_0) & & \delta_p & = & \cos(\omega_0 0) \quad \cos(\omega_0 1) \quad \cdots \quad \cos(\omega_0 (R-1)) \quad 0 \\
A_d(\omega_1) & & -\delta_p & & \cos(\omega_1 0) \quad \cos(\omega_1 1) \quad \cdots \quad \cos(\omega_1 (R-1)) \quad 0 \\
\vdots & & \vdots & & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
A_d(\omega_p) & + & \pm\delta_p & = & \cos(\omega_p 0) \quad \cos(\omega_p 1) \quad \cdots \quad \cos(\omega_p (R-1)) \quad 0 \\
A_d(\omega_s) & & 0 & & \cos(\omega_s 0) \quad \cos(\omega_s 1) \quad \cdots \quad \cos(\omega_s (R-1)) \quad 1 \\
\vdots & & \vdots & & \vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
A_d(\omega_R) & & 0 & & \cos(\omega_R 0) \quad \cos(\omega_R 1) \quad \cdots \quad \cos(\omega_R (R-1)) \quad \pm 1
\end{array}
\begin{array}{c}
\\ a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(R-1) \\ \delta_s
\end{array}.$$

Iteration of this equation will keep the pass band ripple δ_p fixed and minimize the stop band ripple δ_s. A problem with convergence occurs if one of the δ's becomes negative during the iterations. A modification to the basic exchange has been developed to give reliable convergence [link].

The Hofstetter, Oppenheim, and Siegel Algorithm

This algorithm [link], [link], [link] came into existence in order to design the filters posed by Herrmann and Schüssler [link], [link] where both the pass and stop band ripple sizes, δ_p and δ_s, are fixed and the location of the transition band is not directly controlled. This problem results in a maximum ripple design which, for the lowpass filter, requires extremal frequencies at both $\omega = 0$ and $\omega = \pi$ but does not use either pass or stop band frequencies ω_p or ω_s. This results in R extremal frequencies giving R equations to find the R values of $a(n)$.

Equation:

$$\begin{array}{ccccc}
A_d(\omega_0) & & \delta_p & = & \cos(\omega_0 0) \quad \cos(\omega_0 1) \quad \cdots \quad \cos(\omega_0 (R-1)) \\
A_d(\omega_1) & & -\delta_p & & \cos(\omega_1 0) \quad \cos(\omega_1 1) \quad \cdots \quad \cos(\omega_1 (R-1)) \\
\vdots & & \vdots & & \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
A_d(\omega_{p-1}) & + & \pm\delta_p & = & \cos(\omega_{p-1} 0) \quad \cos(\omega_{p-1} 1) \quad \cdots \quad \cos(\omega_{p-1} (R-1)) \\
A_d(\omega_{s+1}) & & \delta_s & & \cos(\omega_{s+1} 0) \quad \cos(\omega_{s+1} 1) \quad \cdots \quad \cos(\omega_{s+1} (R-1)) \\
\vdots & & \vdots & & \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
A_d(\omega_{R-1}) & & \pm\delta_s & & \cos(\omega_{R-1} 0) \quad \cos(\omega_{R-1} 1) \quad \cdots \quad \cos(\omega_{R-1} (R-1))
\end{array}
\begin{array}{c}
\\ a(0) \\ a(1) \\ a(2) \\ \vdots \\ a(R-1)
\end{array}.$$

This algorithm is iterated as a multiple exchange, keeping the number of ripples in the pass and stop band constant, to give an optimal extra ripple filter. The location and width of the transition band is controlled only by the choice of how the number of initial ripples are divided between the pass and stop band. The final filter may not have the transition located where you want it. Indeed, no solution may exist with the desired location of the transition band.

The designs produced by the HOS algorithm are always maximum ripple but this comes with a loss of accurate control over the location of the transition band. The algorithm is not, strictly speaking, an optimization algorithm. It is an interpolation algorithm. The Chebyshev error is not minimized, the designed amplitude interpolates the specified error ripples. However, although not directly minimized, the transition band width of these designs seems to be minimized [link], [link], [link]. Extra or maximum ripple designs seem to be efficient in using all the zeros to produce small ripple size and

narrow transition bands, however, the loss of accurate control over the location of the transition bands becomes even more problematic with multiple transition band designs. Perhaps some compromise methods can be devised that use some of the efficiency of the maximum ripple approximations with some of the control of other methods. The next two design methods are of that type.

The Shpak and Antoniou Algorithm

Shpak and Antoniou [link] propose decoupling the size of the pass and stopband ripple sizes in order to have control over the pass and stop band edges and have an extra ripple design. The Parks-McClellan design has the ripple sizes related with a fixed weight $\delta_p = K\,\delta_s$, the modified Parks-McClellan design fixes one ripple size and minimizes the other, the Hoffstetter, Oppenheim, and Siegel design fixes both ripple sizes but cannot set the transition band edges. The Shpak-Antoniou design fixes the transition band edges and gives a maximum ripple design with minimum ripple but the relationship of the pass and stopband ripple is uncontrolled.

This method has two ripple sizes, δ_p and δ_s, appended to the $a(n)$ vector similar to the single δ used in [link] or [link]. This allows controlling an additional extremal frequency and results in an extra ripple approximation. This can become somewhat complicated for multiple transition bands but seems very flexible [link].

The New Equal Ripple Design Formulation and Exchange Algorithm

Because the arguments in the Weisburn, Parks, and Shenoy paper [link] require the assumption of no signal or noise energy in the transition band, it is now more obvious that a narrow transition band is very desirable. For this reason it may be better to fix the pass and stop band peak error, δ_p and δ_s and the transition band center frequency ω_o then minimize the transition band width rather than fixing the pass and stop band edges, ω_p and ω_s, then minimizing δ_p and δ_s. Two methods have been recently developed to address this point of view. The first is a new exchange algorithm that is in some ways a combination of the Parks-McClellan and Hofstetter-Oppenheim-Segiel algorithms [link] and the second is a limiting case for a constrained least squares method based on Lagrange multipliers [link], [link], [link], [link] using tight constraints.

For problems where the signal and noise spectra are such that a specific frequency ω_o that separates the desired passband from the desired stopband can be specified but specific separate transition band edges, $\omega_p < \omega_s$, cannot, we formulate [link] a design method where the pass and stop band ripple sizes, δ_p and δ_s are specified along with the separation frequency, ω_o. The algorithm described below will interpolate the specified ripple sizes exactly (as the HOS algorithm does) but will allow exact control over the location of ω_o by not requiring maximum ripple. Although not set up to be an optimization procedure, it seems to minimize the transition band width. This formulation suits problems where there is no obvious transition band (“don't care band") having no signal or noise energy to be passed or rejected.

The optimal Chebyshev filter designed with this new algorithm is generally not extra ripple and, therefore, will have an extremal frequency at $\omega = 0$ or $\omega = \pi$ as the Parks-McClellan formulation does. Because we are trying to minimizing the transition band width, we do not specify both the edges, ω_p and ω_s, but only one of them or, perhaps, the center of the transition band, ω_o. This results in R equations which are used to find the R coefficients $a(n)$. The equations are formulated by adding the alternating peak pass and stop band ripples to the A_d in [link] and not having the special last column of

C nor the unknown δ appended to a as was done by Parks and McClellan in [link]. The resulting equation to be iterated in our new exchange algorithm has the form

Equation:

$$\begin{array}{ccccccccc} A_d(\omega_0) & & \delta_p & & \cos(\omega_0 0) & \cos(\omega_0 1) & \cdots & \cos(\omega_0(R-1)) & \\ A_d(\omega_1) & & -\delta_p & & \cos(\omega_1 0) & \cos(\omega_1 1) & \cdots & \cos(\omega_1(R-1)) & a(0) \\ \vdots & & \vdots & & \vdots & & & \vdots & a(1) \\ A_d(\omega_o) & + & 0 & = & \cos(\omega_o) & \cos(\omega_o 1) & \cdots & \cos(\omega_o(R-1)) & a(2) \\ A_d(\omega_{s+1}) & & \delta_s & & \cos(\omega_{s+1} 0) & \cos(\omega_{s+1} 1) & \cdots & \cos(\omega_{s+1}(R-1)) & \vdots \\ \vdots & & \vdots & & \vdots & & & \vdots & a((R-1)) \\ A_d(\omega_{R-1}) & & \pm\delta_s & & \cos(\omega_{R-1} 0) & \cos(\omega_{R-1} 1) & \cdots & \cos(\omega_{R-1}(R-1)) & \end{array} .$$

The exchange algorithm is done as by Parks and McClellan finding new extremal frequencies at each iteration, but with fixed ripple sizes in both pass and stop bands. This new algorithm reduces the transition band width as done by the Hofstetter, Oppenheim, and Siegel method but with the transition band location controlled and without requiring the extra ripple solution. Note that any transition band frequency could be fixed. It could be $A_d(\omega_o) = 1/2$ to fix the half-power point. It could be $A_d(\omega_p) = 1 - \delta_p$ to fix the pass band edge. Or it could be $A_d(\omega_s) = \delta_s$ to fix the stop band edge.

Extending this formulation and algorithm to the multiple transition band case complicates the problem as the solution may not be unique or may have anomalous behavior in one of the transition bands. Details of the solution to this problem are given in [link].

Estimations of , the Length of Optimal Chebyshev FIR Filters

All of the design methods discussed so far have assumed that N,the length of the filter, is given as part of the secifications. In many cases, perhaps even most, N is a parameter that we would like to minimize. Often specifications are to meet certain pass and stopband ripple specifications with given pass and stopband edges and with the shortest possible filter. None of our methods will do that. Indeed, it is not clear how to do that kind of optimization other than by some sort of search. In other words, design a set of filters of different lengths and choose the one that meet the specifications with minimum length.

Fortunately, emperical formulas have been derived that give a good estimate of the relationship of the length of an optimal Chebyshev FIR filter for given pass and stopband ripple and transition band edges [link], [link]. Kaiser's formula is

Equation:

$$N = \frac{-20\log_{10}\left(\sqrt{\delta_p \delta_s}\right) - 13}{14.6(f_s - f_p)} + 1$$

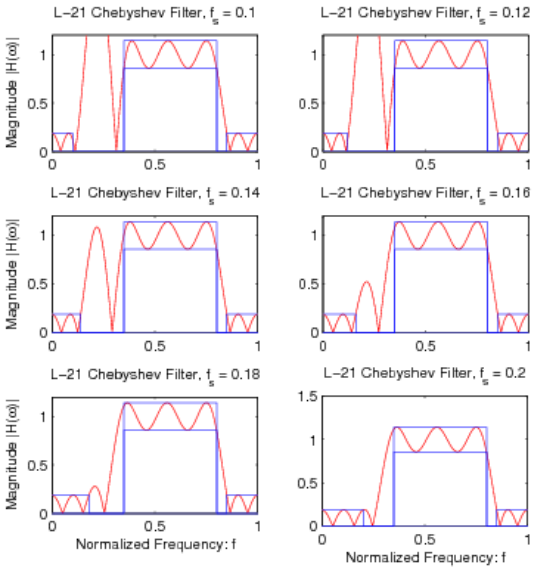
and it is fairly accurate for average filter specifications (neither wide nor narrow bands).

Examples of Optimal Chebyshev Filters

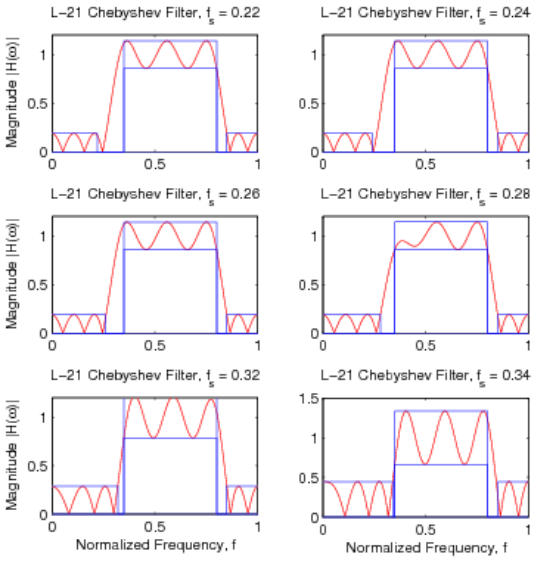
In order to better understand the nature of an optimal Chebyshev and to see the power of the Parks-McClellan algorithm, we present the design of a length-21 linear phase FIR bandpass filter. To see the effects of the design specifications, we will fix the two pass band edges and the upper stop band edge, then look at the effects of varying the lower stop band edge. The Matlab program that generated the designs is:

```
% ChebyPlot9.m generates Chebyshev figures.
% Change in opt frequency response as band edge is changed, csb
1/26/07
N = 20;
M = [0 0 1 1 0 0];
W = [7.5 10 7.5];
ff = [0:512]/512; k=0;
%for fk = .10:.02:.34
%    k = k+1;
clf;
for k = 1:6
    fk = .1 + .02*(k-1);
    F = [0 fk .35 .8 .85 1];
    b = firpm(N,F,M,W);
    %clf;
    axis([0 1 0 1.2]);
    AA = abs(fft(b,1024)); AA = AA(1:513);
    dd = max(AA(1:50));
    ddd = dd*(W(1)/W(2));
    subplot(3,2,k); plot(ff,AA,'r'); hold;
    plot([0 F(2) F(2) F(5) F(5) 1],[dd dd 0 0 dd dd],'b');
    plot([0 F(3) F(3) F(4) F(4) 1],[0 0 1-ddd 1-ddd 0 0],'b');
    plot([0 F(3) F(3) F(4) F(4) 1],[0 0 1+ddd 1+ddd 0 0],'b');
    title('L-21 Chebyshev Filter, f_s = 0.1');
    ylabel('Magnitude |H(\omega)|');
    pause;
end; hold off;
```

The results are shown in Figures [link] and [link].



Amplitude Response of Length-21 Optimal Chebyshev Bandpass Filter with various Stop Band Edges



Amplitude Response of Length-21 Optimal Chebyshev Bandpass Filter with various Stop Band Edges

Note the large transmission peaks in the transition band of Figures [link]a, b, and c that result from the two transition bands being very different in width. As the lower transition band narrows, this peak grows smaller and eventually disappears in [link]f. Note that there are two extremal points in the lower stop band of [link]b and seven in the pass band, while there are three in the lower stop band of Figures [link]c and d and six in the pass band. But, there are always twelve total (thirteen for a case between Figures [link]b and c). In [link]d, there are only five extremal points in the pass band but twelve total. The same filter is optimal for the conditions given in Figures [link]a, b, and c. Much can be learned about optimal filters by running experiments in Matlab. Remember, all of these are optimal for the specifications given.

Chebyshev Approximation using Approximation

It is possible to approximate the effects of Chebyshev approximation by minimizing the p^{th} power of the error. For large p this is close to the results of a true Chebyshev approximation. This is a variation on a method called Lawson's method. This approach is described in [link], [link], [link] using the iterative reweighted least squared (IRLS) error method and looks attractive in that it can use different p in different frequency bands. This would allow, for example, a least squared error approximation in the passband and a Chebyshev approximation in the stopband. The IRLS method can also be used for complex Chebyshev approximations [link].

Characteristics of Optimal Chebyshev Filters

Examples of expected and unexpected results of optimality. Rabiner's work will be used here. The non-unique designs for certain multiband designs will be illustrated.

Complex Chebyshev Approximation

Algorithms that directly use the alternation theorem, such as the standard Remez multiple exchange algorithm, are difficult to apply to the complex approximation or 2-D approximation problem because the concept of “alternation" is difficult to define and the number of ripples in an optimal solution is more difficult to determine [link], [link], [link], [link], [link], [link], [link]. Work has been done on the complex approximation problem at Rice by Parks and Chen [link] and by Burrus, Barreto, and Selesnick [link], [link], at Erlangen by Schuessler, Preuss, Schulist, and Lang [link], [link], [link], [link], [link], [link], at MIT by Alkhairy et al [link], [link], at USC by Tseng and Griffiths [link], [link], at Georgia Tech by Karam and McClellan [link], at Cornell by Burnside and Parks [link], and by Potchinkov and Reemtsen at Cottbus [link], [link], [link], [link], [link]. The work done by Adams which uses an implementation of a constrained quadratic programming algorithm might be useful here [link], [link]. Lang has extended and further developed this constrained approach [link], [link], [link] and Selesnick is applying it to IIR filter design [link]. Tseng gives a good summary of complex approximation in [link].

Conclusions and Discussions of Chebyshev Design

By adding the Chebyshev filter design methods described above to the Parks-McClellan algorithm, one has a rather complete set of approaches to equal ripple filter designs that allows a wide variety of specifications. The new exchange algorithm which minimizes the transition band width while allowing the specification of the center or either edge of the transition band edge may fit many design environments better than the traditional Parks-McClellan. An alternative approach which specifies the pass and stop band peak error yet has **no** zero weighted transition band will be presented in Constrained

Least Squares Design[link], [link]. Matlab programs are available for the Parks-McClellan algorithm, the modified Parks-McClellan algorithm, the Hofstetter-Oppenheim-Siegel algorithm, the new minimum transition band design algorithm, and the constrained least squares algorithm. They are written with a common format and notation to easily see how they are programmed and how they are related. This book generally presents the lowpass case. The bandpass and multi-band cases use the same ideas but are a bit more complicated and are discussed in more detail in the references.

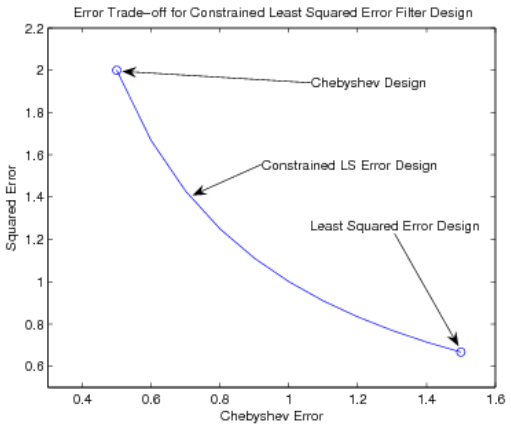
Taylor Series, Maximally Flat, and Zero Moment Design Criteria

The third major approximation criterion uses some measure of the smoothness or flatness of the frequency response. Work has been done by Herrmann [link], P. P. Vaidyanathan, and Selesnick and Burrus [link], [link], [link], [link]. This approach is related to how polynomial signals are processed and may be related to zero moments in wavelet systems.

Constrained Approximation and Mixed Criteria

Trade-off of Error Measures and Design Specifications

In many filter design problems, more than one criterion is important. For example, both L_2 and L_∞ may be of interest in one filter. Often one is posed as a constraint and the other as an optimized variable. Indeed, because L_2 approximation minimizes the error energy and because Parseval's theorem states that an optimal L_2 frequency domain approximation is also an optimal L_2 time domain approximation, an L_∞ constrained minimum L_2 error approximation seems a good practical approach. To see how this might have advantages, it is informative to examine the relationship of the L_2 error to the L_∞ error as the constraint is varied from tight to loose [link], [link] in [link]. From this one can see just how sensitive one error is to the other and how the traditional designs are extremes on this diagram.



The Squared Error vs. the Chebyshev Error for the Constrained Least Squared Error FIR Filter

Another trade-off is the error in a Chebyshev design as a function of the transition band location. There are certain locations of transition band or band edges that give much lower ripple size than others. Rabiner has examined that relation [link], [link].

Constrained Least Squares Design

There are problems where the peak error or Chebyshev error is important. This can be minimized directly using the Remez exchange algorithm but, in many cases, is better controlled by use of a peak error constraint on the basic least squared error formulation of the problem [link], [link], [link], [link]. An efficient algorithm for minimizing the constrained least squared error uses Lagrange multipliers [link], [link] and the Kuhn-Tucker conditions [link], [link].

Similar to the Chebyshev design problem, there are two formulations of the problem: one where there is a well defined transition band separating the desired signal spectrum (passband) from the noise or interfering signal spectrum (stopband) and the second where there is a well defined frequency that separates the pass and stopband but no well defined transition band.

The first case would include situations with signals residing in specified bands separated by “guard bands" such as commercial radio and TV transmissions. It also includes cases where due to multirate sampling, certain well

defined bands are aliased into other well defined bands. The Parks-McClellan and Shpak-Antoniou Chebyshev designs address this case for the Chebyshev error. Adams' method [link], [link], [link], [link], [link], [link] described below applies to the constrained least squares design with a specified transition band.

The second case would include signals with known spectral support with additive white or broad-band noise. In these cases there is no obvious transition band or “don't care" band. The Hoffstetter-Oppenheim-Siegel and the method of the section Chebyshev Approximations using the Exchange Algorithms address this case for a Chebyshev design. The method in section below applies to the constrained least squares design [link] without a specified transition band.

The Lagrangian

To pose the constrained least squared error optimization problem, we use a Lagrange multiplier formulation. First define the Lagrangian as
Equation:

$$\mathscr{L} \;=\; P\int_0^{\pi}\left(A\left(\omega\right)-A_d\left(\omega\right)\right)^2 d\omega+\sum_i \mu_i\left(A\left(\omega_i\right)-\left[A_d\left(\omega_i\right)\pm T\left(\omega_i\right)\right]\right)$$

where the μ_i are the necessary number of Langrange multipliers and P is a scale factor that can be chosen for simplicity later. The first term in [link] is the integral squared error of the frequency response to be minimized and the second term will be zero when the equality constraints are satisfied at the frequencies, ω_i. The function $T(\omega)$ is the constraint function in that $A(\omega)$ must satisfy
Equation:

$$A_d\left(\omega\right)+T\left(\omega\right)\geq A\left(\omega\right)\geq A_d\left(\omega\right)-T\left(\omega\right).$$

Necessary conditions for the minimization of the integral squared error are that the derivative of the Lagrangian with respect to the filter parameters $a(n)$ defined in Equation 49 from FIR Digital Filters and to the Lagrange multipliers μ_i be zero [link].

The derivatives of the Lagrangian with respect to $a(n)$ are
Equation:

$$\frac{d\mathscr{L}}{d\,a(n)} \;=\; P\int_0^{\pi} 2\,\left(A\left(\omega\right)-A_d\left(\omega\right)\right)\frac{dA}{da}\,d\omega \;+\; \sum_i \mu_i \frac{dA}{da}\bigg|_{\omega_i}$$

where from Equation 49 from FIR digital Filters we have for $n=1,2,\cdots,M$
Equation:

$$\frac{d\,A(\omega)}{d\,a(n)} \;=\cos\left(\omega n\right)$$

and for $n=0$
Equation:

$$\frac{d\,A(\omega)}{d\,a(0)} \;=\; K.$$

For $n = 1, 2, \cdots, M$ this gives
Equation:

$$\frac{d\mathscr{L}}{d\,a(n)} \;=\; 2\,P\left[\int A\left(\omega\right)\;\cos\left(\omega n\right)d\omega - \int A_d\left(\omega\right)\;\cos\left(\omega n\right)d\omega\right] + \sum_i \mu_i\;\cos\left(\omega_i n\right)$$

and for $n = 0$ gives
Equation:

$$\frac{d\mathscr{L}}{d\,a(0)} \;=\; 2\,P\,K\left[\int A\left(\omega\right)d\omega - \int A_d\left(\omega\right)d\omega\right] + \sum_i \mu_i\,K.$$

Using Equation 50 from FIR Digital Filters for $n = 1, 2, \cdots, M$, we have
Equation:

$$\frac{d\mathscr{L}}{d\,a(n)} \;=\; \pi\,P[a\left(n\right) - a_d\left(n\right)] + \sum_i \mu_i\;\cos\left(\omega_i n\right) = 0$$

and for $n = 0$
Equation:

$$\frac{d\mathscr{L}}{d\,a(0)} \;=\; 2\pi\,P\,K^2[a\left(0\right) - a_d\left(0\right)] + K\sum_i \mu_i \;=\; 0.$$

Choosing $P = 1/\pi$ gives
Equation:

$$a\left(n\right) \;=\; a_d\left(n\right) - \sum_i \mu_i\;\cos\left(\omega_i n\right)$$

and
Equation:

$$a\left(0\right) \;=\; a_d\left(0\right) - \frac{1}{2K}\sum_i \mu_i$$

Writing [link] and [link] in matrix form gives
Equation:

$$\mathbf{a} \;=\; \mathbf{a_d} - \mathbf{H}\,\mu.$$

where $\mathbf{H}$ is a matrix with elements
Equation:

$$h\left(n, i\right) \;= \cos\left(\omega_i n\right)$$

except for the first row which is

Equation:

$$h\left(0,i\right) \;=\; \frac{1}{2K}$$

because of the normalization of the $a(0)$ term. The $a_d\left(n\right)$ are the cosine coefficients for the unconstrained approximation to the ideal filter which result from truncating the inverse DTFT of $A_d\left(\omega\right)$.

The derivative of the Lagrangian in [link] with respect to the Lagrange multipliers μ_i, when set to zero, gives
Equation:

$$A\left(\omega_i\right) \;=\; A_d\left(\omega_i\right) \pm T\left(\omega_i\right) \;=\; A_c\left(\omega_i\right)$$

which is simply a statement of the equality constraints.

In terms of the filter's cosine coefficients $a(n)$, from Equation 49 from FIR Digital Filters, this can be written"
Equation:

$$A_c\left(\omega_i\right) \;=\; \sum_n a\left(n\right)\;\cos\left(\omega_i n\right) + K\,a\left(0\right)$$

and as matrices
Equation:

$$\mathbf{A_c} \;=\; \mathbf{G}\,\mathbf{a}$$

where $\mathbf{A_c}$ is the vector of frequency response values which are the desired response plus or minus the constraints evaluated at the frequencies in the constraint set. The frequency response must interpolate these values. The matrix $\mathbf{G}$ is
Equation:

$$g\left(i,n\right) \;=\cos\left(\omega_i n\right)$$

except for the first column which is
Equation:

$$g(i,0) \;=\; K.$$

Notice that if $K = 1/\sqrt{2}$, the first rows and columns are such that we have $\mathbf{G^T} = \mathbf{H}$.

The two equations [link] and [link] that must be satisfied can be written as a single matrix equation of the form
Equation:

$$\begin{bmatrix} \mathbf{I} & \mathbf{H} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{a}_d \\ \mathbf{A_c} \end{bmatrix}$$

or, if $K = 1/\sqrt{2}$, as
Equation:

$$\begin{bmatrix} \mathbf{I} & \mathbf{G}^T \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{a}_d \\ \mathbf{A_c} \end{bmatrix}$$

which have as solutions
Equation:

$$\begin{aligned} \mu &= (\mathbf{GH})^{-1} (\mathbf{G}\,\mathbf{a_d} - \mathbf{A_c}) \\ \mathbf{a} &= \mathbf{a}_d - \mathbf{H}\mu \end{aligned}$$

The filter corresponding to the cosine coefficients $a(n)$ minimize the L_2 error norm subject the equality conditions in [link].

Notice that the term in [link] of the form $\mathbf{G}\,\mathbf{a_d}$ is the frequency response of the optimal unconstrained filter evaluated at the constraint set frequencies. Equation [link] could, therefore, be written
Equation:

$$\mu = (\mathbf{G}\,\mathbf{H})^{-1} (\mathbf{A_u} - \mathbf{A_c})$$

The Constrained Weighted Least Squares Design of FIR Filters

Combining the weighted least squared error formulation with the constrained least squared error gives the general formulation of this class of problems.

We now modify the Lagrangian in [link] to allow a weighted squared error giving
Equation:

$$\mathscr{L} = \frac{1}{\pi} \int_0^\pi W(\omega) (A(\omega) - A_d(\omega))^2 \, d\omega + \sum_i \mu_i (A(\omega_i) - A_d(\omega_i) \pm T(\omega_i))$$

with a corresponding derivative of
Equation:

$$\frac{d\mathscr{L}}{d\,a(n)} = \frac{2}{\pi} \int (W(\omega) (A(\omega) - A_d(\omega)) \frac{dA}{da} \, d\omega + \sum_i \mu_i \frac{dA}{da}\bigg|_{\omega_i}$$

The integral cannot be carried out analytically for a general weighting function, but if the weight function is constant over each subband, Equation 47 from Least Squared Error Design of FIR Filters can be written
Equation:

$$\frac{d\mathscr{L}}{d\,a(n)} = \frac{2}{\pi} \sum_k \int_{\omega_k}^{\omega_{k+1}} \left(W_k \left(\sum_{m=1}^{M} a(m) \, \cos(\omega m) + K\,a(0) - A_d(\omega) \right) \right) \cos(\omega n) \, d\omega + \sum_i \mu_i \frac{dA}{da}\bigg|_{\omega_i}$$

which after rearranging is
Equation:

$$= \sum_{m=1}^{M} \left[\frac{2}{\pi} \sum_{k} W_k \int_{\omega_k}^{\omega_{k+1}} (\cos(\omega m)\ \cos(\omega n))\, d\omega \right] a(m)$$

Equation:

$$-\frac{2}{\pi} \sum_{k} W_k \int_{\omega_k}^{\omega_{k+1}} A_d(\omega)\ \cos(\omega n)\, d\omega + \sum_{i} \mu_i \cos(\omega_i n) = 0$$

where the integral in the first term can now be done analytically. In matrix notation Equation 49 from Least Squared Error Design of FIR Filters is
Equation:

$$\mathbf{R}\,\mathbf{a} - \mathbf{a_{d_w}} + \mathbf{H}\,\mu = \mathbf{0}$$

This is a similar form to that in the multiband paper where the matrix $\mathbf{R}$ gives the effects of weighting with elements
Equation:

$$r(n,m) = \frac{2}{\pi} \sum_{k} W_k \int_{\omega_k}^{\omega_{k+1}} (\cos(\omega m)\ \cos(\omega n))\, d\omega$$

except for the first row which should be divided by $2K$ because of the normalizing of the $a(0)$ term in Equation 49 from FIR Digital Filters and [link] and the first column which should be multiplied by K because of Equation 51 from FIR Digital Filters and [link]. The matrix $\mathbf{R}$ is a sum of a Toeplitz matrix and a Hankel matrix and this fact might be used to advantage and $\mathbf{a_{d_w}}$ is the vector of modified filter parameters with elements
Equation:

$$a_{d_w}(n) = \frac{2}{\pi} \sum W_k \int_{\omega_k}^{\omega_{k+1}} A_d(\omega)\ \cos(\omega n)\, d\omega$$

and the matrix $\mathbf{H}$ is the same as used in [link] and defined in [link]. Equations Equation 50 from Least Squared Error Design of FIR Filters and [link] can be written together as a matrix equation
Equation:

$$\begin{bmatrix} \mathbf{R} & \mathbf{H} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{d_w} \\ \mathbf{A}_c \end{bmatrix}$$

The solutions to Equation 50 from Least Squared Error Design of FIR Filters and [link] or to [link] are
Equation:

$$\mu = \left(\mathbf{G}\mathbf{R}^{-1}\mathbf{H}\right)^{-1} \left(\mathbf{G}\mathbf{R}^{-1}\mathbf{a_{d_w}} - \mathbf{A_c}\right)$$

Equation:

$$\mathbf{a} = \mathbf{R}^{-1}\left(\mathbf{a}_{d_w} - \mathbf{H}\,\mu\right)$$

which are ideally suited to a language like Matlab and are implemented in the programs at the end of this book.

Since the solution of $\mathbf{R}\,\mathbf{a_u} = \mathbf{a_{d_w}}$ is the optimal unconstrained weighted least squares filter, we can write [link] and [link] in the form

Equation:

$$\mu \;=\; \left(\mathbf{G}\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\mathbf{G}\,\mathbf{a_u} - \mathbf{A_c}\right) \;=\; \left(\mathbf{G}\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\left(\mathbf{A_u} - \mathbf{A_c}\right)$$

Equation:

$$\mathbf{a} \;=\; \mathbf{a}_u - \mathbf{R}^{-1}\mathbf{H}\,\mu$$

The Exchange Algorithms

This Lagrange multiplier formulation together with applying the Kuhn-Tucker conditions are used in an iterative multiple exchange algorithm similar to the Remez exchange algorithm to give the complete design method.

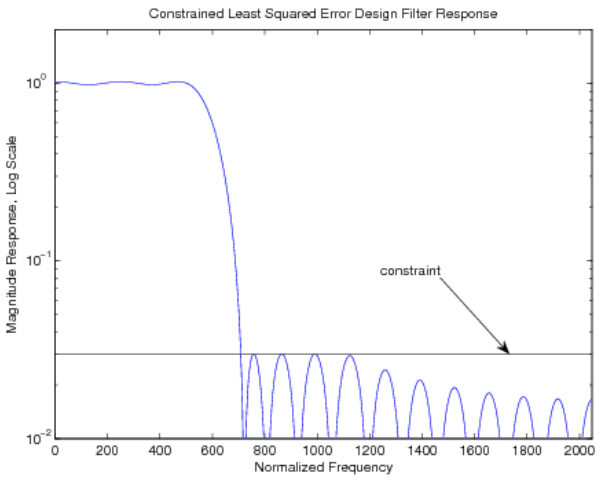
One version of this exchange algorithm applies to the problem posed by Adams with specified pass and stopband edges and with zero error weighting in the transition band. This problem has the structure of a quadratic programming problem and could be solved using general QP methods but the multiple exchange algorithm suggested here is probably faster.

The second version of this exchange algorithm applies to the problem where there is no explicitly specified transition band. This problem is not strictly a quadratic programming problem and our exchange algorithm has no proof of convergence (the HOS algorithm also has no proof of convergence). However, in practice, this program has proven to be robust and converges for a wide variety of lengths, constraints, weights, and band edges. The performance is completely independent of the normalizing parameter K. Notice that the inversion of the $\mathbf{R}$ matrix is done once and does not have to be done each iteration. The details of the program are included in the filter design paper and in the Matlab program at the end of this book.

As mentioned earlier, this design problem might be addressed by general constrained quadratic programming methods [link], [link], [link], [link], [link], [link], [link], [link], [link], [link].

Examples and Observations on CLS Designs

Here we show that the CLS FIR filter design approach is probably the best general FIR filter design method. For example, a length-31 linear phase lowpass FIR filter is designed for a band edge of 0.3 and the constraint that the response in the stop cannot be greater than 0.03 is illustrated in [link].



Response of a Constrained Least Squared Error Filter Design

This filter was designed using the Matlab command: '`fircls1()`' function.

L^p Approximation and the Iterative Reweighted Least Squares Method

We now consider the general L_p approximation which contains the least squares L_2 and the Chebyshev L_∞ cases. This approach is described in [link], [link], [link] using the iterative reweighted least squared (IRLS) alorithm and looks attractive in that it can use different p in different frequency bands. This would allow, for example, a least squared error approximation in the passband and a Chebyshev approximation in the stopband. The IRLS method can also be used for complex Chebyshev approximations [link] and constrained L_2 approximatin.

Iterative Reweighted Least Squares Filter Design Methods

There are cases where it is desirable to design an FIR filter that will minimize the L_p error norm. The error is defined by
Equation:

$$q = \int_\Omega |A(\omega) - A_d(\omega)|^p \, d\omega$$

but we usually work with Q^2. For large p, the results are essentially the same as the Chebyshev filter and this gives a continuum of design between L_2 and L_∞. It also allows the very interesting important possibility of allowing $p(\omega)$ to be a function of frequency. This means one could have different error criteria in different frequency bands. It can be modified to give the same effects as a constraint. This approach is discussed in [link]. It can be applied to complex approximation and to two-dimensional filter design [link], [link].

The least squared error and the minimum Chebyshev error criteria are the two most commonly used linear-phase FIR filter design methods [link]. There are many situations where better total performance would be obtained with a mixture of these two error measures or some compromise design that would give a trade-off

between the two. We show how to design a filter with an L_2 approximation in the passband and a Chebyshev approximation in the stopband. We also show that by formulating the L_p problem we can solve the constrained L_2 approximation problem [link].

This section first explores the minimization of the p^{th} power of the error as a general approximation method and then shows how this allows L_2 and L_∞ approximations to be used simultaneous in different frequency bands of one filter and how the method can be used to impose constraints on the frequency response. There are no analytical methods to find this approximation, therefore, an iterative method is used over samples of the error in the frequency domain. The methods developed here [link], [link] are based on what is called an **iterative reweighted least squared** (IRLS) error algorithm [link], [link], [link] and they can solve certain FIR filter design problems that neither the Remez exchange algorithm nor analytical L_2 methods can.

The idea of using an IRLS algorithm to achieve a Chebyshev or L_∞ approximation seems to have been first developed by Lawson [link] and extended to L_p by Rice and Usow [link], [link]. The basic IRLS method for L_p was given by Karlovitz [link] and extended by Chalmers, et. al. [link], Bani and Chalmers [link], and Watson [link]. Independently, Fletcher, Grant and Hebden [link] developed a similar form of IRLS but based on Newton's method and Kahng [link] did likewise as an extension of Lawson's algorithm. Others analyzed and extended this work [link], [link], [link], [link]. Special analysis has been made for $1 \leq p < 2$ by [link], [link], [link], [link], [link], [link], [link] and for $p = \infty$ by [link], [link], [link], [link], [link], [link]. Relations to the Remez exchange algorithm [link], [link] were suggested by [link], to homotopy [link] by [link], and to Karmarkar's linear programming algorithm [link] by [link], [link]. Applications of Lawson's algorithm to complex Chebyshev approximation in FIR filter design have been made in [link], [link], [link], [link] and to 2-D filter design in [link]. Reference [link] indicates further results may be forthcoming. Application to array design can be found in [link] and to statistics in [link].

This paper unifies and extends the IRLS techniques and applies them to the design of FIR digital filters. It develops a framework that relates all of the above referenced work and shows them to be variations of a basic IRLS method modified so as to control convergence. In particular, we generalize the work of Rice and Usow on Lawson's algorithm and explain why its asymptotic convergence is slow.

The main contribution here is a new robust IRLS method [link], [link] that combines an improved convergence acceleration scheme and a Newton based method. This gives a very efficient and versatile filter design algorithm that performs significantly better than the Rice-Usow-Lawson algorithm or any of the other IRLS schemes. Both the initial and asymptotic convergence behavior of the new algorithm is examined and the reason for occasional slow convergence of this and all other IRLS methods is discovered.

We then show that the new IRLS method allows the use of p as a function of frequency to achieve different error criteria in the pass and stopbands of a filter. Therefore, this algorithm can be applied to solve the constrained L_p approximation problem. Initial results of applications to the complex and two-dimensional filter design problem are presented.

Although the traditional IRLS methods were sometimes slower than competing approaches, the results of this paper and the availability of fast modern desktop computers make them practical now and allow exploitation of their greater flexibility and generality.

Minimum Squared Error Approximations

Various approximation methods can be developed by considering different definitions of norm or error measure. Commonly used definitions are L_1, L_2, and Chebyshev or L_∞. Using the L_2 norm, gives the scalar error to minimize

Equation:

$$q = \sum_{k=0}^{L-1} \left| A\left(\omega_k\right) - A_d\left(\omega_k\right) \right|^2$$

or in matrix notation using [link], the error or residual vector is defined by
Equation:

$$q \; = \; \mathbf{C}\,\mathbf{a} - \mathbf{A_d}$$

giving the scalar error of [link] as
Equation:

$$q \; = \; \epsilon^{\mathbf{T}} \epsilon .$$

This can be minimized by solution of the normal equations [link], [link], [link]
Equation:

$$\mathbf{C^T}\,\mathbf{C}\,\mathbf{a} = \mathbf{C^T}\,\mathbf{A_d} .$$

The weighted squared error defined by
Equation:

$$q \; = \; \sum_{k=0}^{L-1} w_k^2 \left| A\left(\omega_k\right) - A_d\left(\omega_k\right) \right|^2 .$$

or, in matrix notation using [link] and [link] causes [link] to become
Equation:

$$q = \epsilon^{\mathbf{T}}\,\mathbf{W^T}\,\mathbf{W}\,\epsilon$$

which can be minimized by solving
Equation:

$$\mathbf{W}\,\mathbf{C}\,\mathbf{a} = \mathbf{W}\,\mathbf{A_d}$$

with the normal equations
Equation:

$$\mathbf{C^T}\,\mathbf{W^T}\,\mathbf{W}\,\mathbf{C}\,\mathbf{a} = \mathbf{C^T}\,\mathbf{W^T}\,\mathbf{W}\,\mathbf{A_d}$$

where W is an L by L diagonal matrix with the weights w_k from [link] along the diagonal. A more general formulation of the approximation simply requires $W^T W$ to be positive definite. Some authors define the weighted error in [link] using w_k rather than w_k^2. We use the latter to be consistent with the least squared error algorithms in Matlab [link].

Solving [link] is a direct method of designing an FIR filter using a weighted least squared error approximation. To minimize the sum of the squared error and get approximately the same result as minimizing the integral of the squared error, one must choose L to be 3 to 10 or more times the length L of the filter being designed.

Iterative Algorithms to Minimize the Error

There is no simple direct method for finding the optimal approximation for any error power other than two. However, if the weighting coefficients w_k as elements of W in [link] could be set equal to the elements in $|A-A_d|$, minimizing [link] would minimize the fourth power of $|A-A_d|$. This cannot be done in one step because we need the solution to find the weights! We can, however, pose an iterative algorithm which will first solve the problem in [link] with no weights, then calculate the error vector ϵ from [link] which will then be used to calculate the weights in [link]. At each stage of the iteration, the weights are updated from the previous error and the problem solved again. This process of successive approximations is called the **iterative reweighted least squared** error algorithm (IRLS).

The basic IRLS equations can also be derived by simply taking the gradient of the p-error with respect to the filter coefficients h or a and setting it equal to zero [link], [link]. These equations form the basis for the iterative algorithm.

If the algorithm is a contraction mapping [link], the successive approximations will converge and the limit is the solution of the minimum L_4 approximation problem. If a general problem can be posed [link], [link], [link] as the solution of an equation in the form
Equation:

$$x = f(x),$$

a successive approximation algorithm can be proposed which iteratively calculates x using
Equation:

$$x_{m+1} = f\left(x_m\right)$$

starting with some x_0. The function $f(\cdot)$ maps x_m into x_{m+1} and, if

$\lim_{m\to\infty} x_m = x_0$ where $x_0 = f\left(x_0\right)$, x_0 is the fixed point of the mapping and a solution to [link]. The trick is to find a mapping that solves the desired problem, converges, and converges fast.

By setting the weights in [link] equal to
Equation:

$$w\left(k\right) = \left|A\left(\omega_k\right) - A_d\left(\omega_k\right)\right|^{(p-2)/2},$$

the fixed point of a convergent algorithm minimizes
Equation:

$$q = \sum_{k=0}^{L-1} \left|A\left(\omega_k\right) - A_d\left(\omega_k\right)\right|^p.$$

It has been shown [link] that weights always exist such that minimizing [link] also minimizes [link]. The problem is to find those weights efficiently.

Basic Iterative Reweighted Least Squares

The basic IRLS algorithm is started by initializing the weight matrix defined in [link] and [link] for unit weights with $W_0 = I$. Using these weights to start, the m^{th} iteration solves [link] for the filter coefficients with

Equation:

$$\mathbf{a_m} = \left[\mathbf{C^T W_m^T W_m C}\right]^{-1} \mathbf{C^T W_m^T W_m A_d}$$

This is a formal statement of the operation. In practice one should not invert a matrix, one should use a sophisticated numerical method [link] to solve the overdetermined equations in [link] The error or residual vector [link] for the m^{th} iteration is found by
Equation:

$$\mathbf{\epsilon_m = C\,a_m - A_d}$$

A new weighting vector is created from this error vector using [link] by
Equation:

$$w_{m+1} = \left|\epsilon_{\mathbf{m}}\right|^{(p-2)/2}$$

whose elements are the diagonal elements of the new weight matrix
Equation:

$$\mathbf{W_{m+1}} = diag\left[w_{m+1}\right].$$

Using this weight matrix, we solve for the next vector of filter coefficients by going back to [link] and this defines the basic iterative process of the IRLS algorithm.

It can easily be shown that the a that minimizes [link] is a fixed point of this iterative map. Unfortunately, applied directly, this basic IRLS algorithm does not converge and/or it has numerical problems for most practical cases [link]. There are three aspects that must be addressed. First, the IRLS algorithm must theoretically converge. Second, the solution of [link] must be numerically stable. Finally, even if the algorithm converges and is numerically stable, it must converge fast enough to be practical.

Both theory and experience indicate there are different convergence problems connected with several different ranges and values of p. In the range $2 \leq p < 3$, virtually all methods converge [link], [link], [link]. In the range $3 \leq p < \infty$, the algorithm diverges and the various methods discussed in this paper must be used. As p becomes large compared to 2, the weights carry a larger contribution to the total minimization than the underlying least squared error minimization, the improvement at each iteration becomes smaller, and the likelihood of divergence becomes larger. For $p = \infty$ we can use to advantage the fact that the optimal approximation solution to [link] is unique but the weights in [link] that give that solution are not. In other words, different matrices W give the same solution to [link] but will have different convergence properties. This allows certain alteration to the weights to improve convergence without harming the optimality of the results [link]. In the range $1 < p < 2$, both convergence and numerical problems exist as, in contrast to $p > 2$, the IRLS iterations are undoing what the underlying least squares is doing. In particular, the weights near frequencies with small errors become very large. Indeed, if the error happens to be zero, the weight becomes infinite because of the negative exponent in [link]. For $p = 1$ the solution to the optimization problem is not even unique. The various algorithms that are presented below are based on schemes to address these problems.

The Karlovitz Method

In order to achieve convergence, a second order update is used which only partially changes the filter coefficients a_m in [link] each iteration. This is done by first calculating the unweighted L_2 approximation filter coefficients using Equation 6 from Chebyshev or Equal Ripple Error Approximation Filters as

Equation:

$$a_0 = \left[C^T\,C\right]^{-1} C^T\,A_d.$$

The error or residual vector Equation 1 from Chebyshev or Equal Ripple Error Approximation Filters for the m^{th} iteration is found as Equation 97 from Sampling, Up-Sampling, Down-Sampling, and Multi-Rate by
Equation:

$$\epsilon_m = C\,a_m - A_d$$

and the new weighting vector is created from this error vector using [link] by
Equation:

$$w_{m+1} = |\epsilon_m|^{(p-2)/2}$$

whose elements are the diagonal elements of the new weight matrix
Equation:

$$W_{m+1} = diag\,[w_{m+1}].$$

This weight matrix is then used to calculate a temporary filter coefficient vector by
Equation:

$$\widehat{a}_{m+1} = \left[C^T W_{m+1}^T W_{m+1} C\right]^{-1} C^T W_{m+1}^T W_{m+1} A_d.$$

The vector of filter coefficients that is actually used is only partially updated using a form of adjustable step size in the following second order linearly weighted sum
Equation:

$$a_{m+1} = \lambda \widehat{a}_{m+1} + (1-\lambda) a_m$$

Using this filter coefficient vector, we solve for the next error vector by going back to [link] and this defines Karlovitz's IRLS algorithm [link].

In this algorithm, λ is a convergence parameter that takes values $0 < \lambda \leq 1$. Karlovitz showed that for the proper λ, the IRLS algorithm using [link] always converges to the globally optimal L_p approximation for p an even integer in the range $4 \leq p < \infty$. At each iteration the L_p error has to be minimized over λ which requires a line search. In other words, the full Karlovitz method requires a multi-dimensional weighted least squares minimization and a one-dimensional p^{th} power error minimization at each iteration. Extensions of Karlovitz's work [link] show the one-dimensional minimization is not necessary but practice shows the number of required iterations increases considerably and robustness in lost.

Fletcher et al. [link] and later Kahng [link] independently derive the same second order iterative algorithm by applying Newton's method. That approach gives a formula for λ as a function of p and is discussed later in this paper. Although the iteration count for convergence of the Karlovitz method is good, indeed, perhaps the best of all, the minimization of λ at each iteration causes the algorithm to be very slow in execution.

Newton's Methods

Both the new method in section 4.3 and Lawson's method use a second order updating of the weights to obtain convergence of the basic IRLS algorithm. Fletcher et al. [link] and Kahng [link] use a linear summation for the updating similar in form to [link] but apply it to the filter coefficients in the manner of Karlovitz rather than the weights as Lawson did. Indeed, using our development of Karlovitz's method, we see that Kahng's method and Fletcher, Grant, and Hebden's method are simply a particular choice of λ as a function of p in Karlovitz's method. They derive
Equation:

$$\lambda = \frac{1}{p-1}$$

by using Newton's method to minimize ε in [link] to give for [link]
Equation:

$$a_m = \left(\widehat{a}_m + (p-2)a_{m-1}\right) / \,(p-1).$$

This defines Kahng's method which he says always converges [link]. He also notes that the summation methods in the sections Calculation of the Fourier Transform and Fourier Series using the FFT, Sampling Functions--the Shah Function, and Down-Sampling, Subsampling, or Decimation do not have the possible restarting problem that Lawson's method theoretically does. Because Kahng's algorithm is a form of Newton's method, its asymptotic convergence is very good but the initial convergence is poor and very sensitive to starting values.

A New Robust IRLS Method

A modification and generalization of an acceleration method suggested independently by Ekblom [link] and by Kahng [link] is developed here and combined with the Newton's method of Fletcher, Grant, and Hebden and of Kahng to give a robust, fast, and accurate IRLS algorithm [link], [link]. It overcomes the poor initial performance of the Newton's methods and the poor final performance of the RUL algorithms.

Rather than starting the iterations of the IRLS algorithms with the actual desired value of p, after the initial L_2 approximation, the new algorithm starts with $p = K*2$ where K is a parameter between one and approximately two, chosen for the particular problem specifications. After the first iteration, the value of p is increased to $p = K^2*2$. It is increased by a factor of K at each iteration until it reaches the actual desired value. This keeps the value of p being approximated just ahead of the value achieved. This is similar to a homotopy where we vary the value of p from 2 to its final value. A small value of K gives very reliable convergence because the approximation is achieved at each iteration but requires a large number of iterations for p to reach its final value. A large value of K gives faster convergence for most filter specifications but fails for some. The rule that is used to choose p_m at the m^{th} iteration is
Equation:

$$p_m = \min\,(p, K\,p_{m-1}).$$

Each iteration of our new variable p method is implemented by the basic algorithm described as Karlovitz's method but using the Newton's method based value of λ from Fletcher or Kahng in [link]. Both Ekblom and Kahng only used $K = 2$ which is too large in almost all cases.

We also tried the generalized acceleration scheme with the basic Karlovitz method and the RUL algorithm. Although it improved the initial performance of the Karlovitz method, the slowness of each iteration still made this method unattractive. Use with the RUL algorithm gave only a minor improvement of initial performance and no improvement of the poor final convergence.

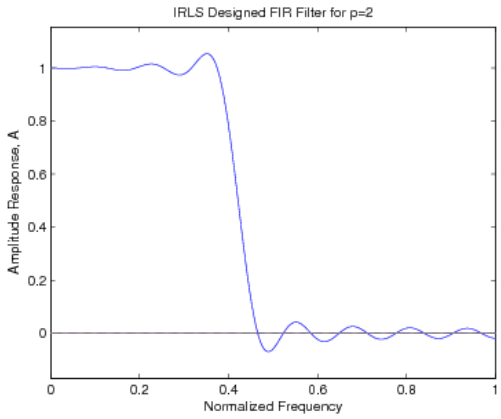
Our new algorithm uses three distinct concepts:

- The basic IRLS which is a straight forward algorithm with linear convergence [link] when it converges.
- The second order or Newton's modification which increases the number of cases where initial convergence occurs and gives quadratic asymptotic convergence [link], [link].
- The controlled increasing of p from one iteration to the next is a modification which gives excellent initial convergence and allows adaptation for “difficult" cases.

The best total algorithm, therefore, combines the increasing of p given in [link] the updating the filter coefficients using [link], and the Newton's choice of λ in [link]. By slowly increasing p, the error surface slowly changes from the parabolic shape of L_2 which Newton's method is based on, to the more complicated surface of L_p. The question is how fast to change and, from experience with many examples, we have learned that this depends on the filter design specifications.

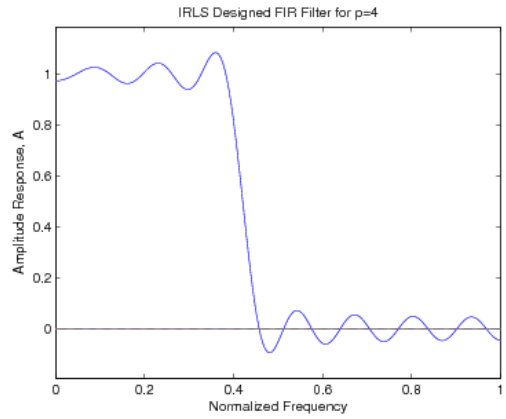
A Matlab program that implements this basic IRLS algorithm is given in the appendix of this paper. It uses an updating of $A(\omega_k)$ in the frequency domain rather than of $a(n)$ in the time domain to allow modifications necessary for using different p in different bands as will be developed later in this paper.

An example design for a length $L = 31$, passband edge $fp = 0.4$, stopband edge $fs = 0.44$, and $p = 2$ the program does not have to iterate and give the response in [link].



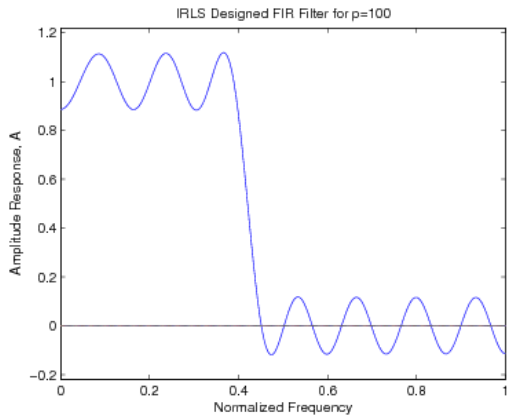
Response of an Iterative Reweighted Least Squares Design with $p = 2$

For the same specifications except $p = 4$ we get [link]



Response of an IRLS Design with $p = 4$

and for $p = 100$ we get [link]



Response of an IRLS Design with $p = 100$

Different Error Criteria in Different Bands

Probably the most important use of the L_p approximation problem posed here is its use for designing filters with different error criteria in different frequency bands. This is possible because the IRLS algorithm allows an error power that is a function of frequency $p(\omega)$ which can allow an L_2 measure in the passband and a Chebyshev error measure in the stopband or any other form. This is important if an L_2 approximation is needed in the passband because Parseval's theorem shows that the time domain properties of the filtered signal will be well preserved but, because of unknown properties of the noise or interference, the stopband attenuation must be less than some specified valued.

The new algorithm described in "A New Robust IRLS Method" was modified so that the iterative updating is done to $A(\omega)$ rather than to $a(n)$. Because the Fourier transform is linear, the updating of [link] can also be

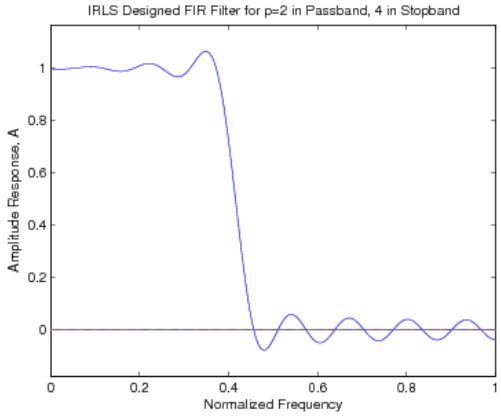
achieved by
Equation:

$$A_{m+1}(\omega) = \lambda \widehat{A}_{m+1}(\omega) + (1-\lambda) A_m(\omega).$$

The Matlab program listed in the appendix uses this form. This type of updating in the frequency domain allows different p to be used in different bands of $A(\omega)$ and different update parameters λ to be used in the appropriate bands. In addition, it allows a different constant K weighting to be used for the different bands. The error for this problem is changed from [link] to be
Equation:

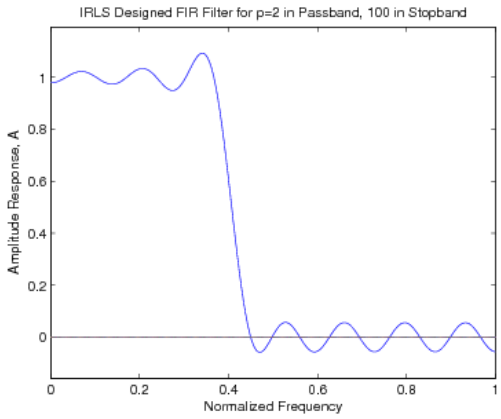
$$q = \sum_{k=0}^{k_0} \left| A(\omega_k) - A_d(\omega_k) \right|^2 + K \sum_{k=k_0+1}^{L-1} \left| A(\omega_k) - A_d(\omega_k) \right|^p$$

[link] shows the frequency response of a filter designed with a passband $p = 2$, a stopband $p = 4$, and a stopband weight of $K = 1$.

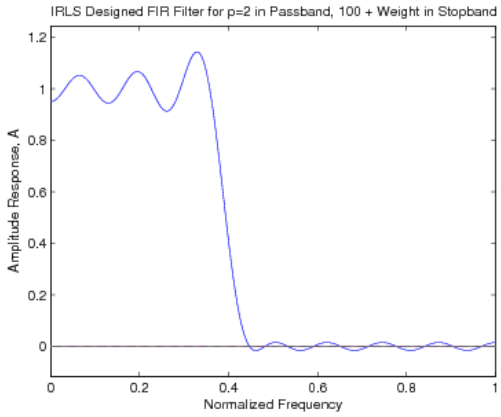


Response of an IRLS Design with $p = 2$ in the Stopband and $p = 4$ in the Stopband

[link] gives the frequency response for the same specifications but with $p = 100$ and [link] adds a constant weight to the stopband.



Response of an IRLS Design with $p = 2$ in the Stopband and $p = 100$ in the Stopband



Response of an IRLS Design with $p = 2$ in the Stopband and $p = 100$ plus a weight in the Stopband

The Constrained Approximation

In some design situations, neither a pure L_2 nor a L_∞ or Chebyshev approximation is appropriate. If one evaluates both the squared error and the Chebyshev error of a particular filter, it is easily seen that for an optimal least squares solution, a considerable reduction of the Chebyshev error can be obtained by allowing a small increase in the squared error. For the optimal Chebyshev solution the opposite is true. A considerable reduction of the squared error can be obtained by allowing a small increase in the Chebyshev error. This suggests a better filter might be obtained by some combination of L_2 and L_∞ approximation. This problem is stated and addressed by Adams [link] and by Lang [link], [link].

We have applied the IRLS method to the constrained least squares problem by adding an error based weighting function to unity in the stopband only in the frequency range where the response in the previous iteration exceeds the constraint. The frequency response of an example is the that was illustrated in [link] as obtained using the CLS algorithm. The IRLS approach to this problem is currently being evaluated and compared to the approach used by Adams. The initial results are encouraging.

Application to the Complex Approximation and the 2D Filter Design Problem

Although described above in terms of a one dimensional linear phase FIR filter, the method can just as easily be applied to the complex approximation problem and to the multidimensional filter design problem. We have obtained encouraging initial results from applications of our new IRLS algorithm to the optimal design of FIR filters with a nonlinear phase response. By using a large p we are able to design essentially Chebyshev filters where the Remez algorithm is difficult to apply reliably.

Our new IRLS design algorithm was applied to the two examples considered by Chen and Parks [link] and by Schulist [link], [link] and Preuss [link], [link]. One is a lowpass filter and the other a bandpass filter, both approximating a constant group delay over their passbands. Examination of magnitude frequency response plots, imaginary vs. real part plots, and group delay frequency response plots for the filters designed by the IRLS method showed close agreement with published results [link]. The use of an L_p approximation may give more desirable results than a true Chebyshev approximation. Our results on the complex approximation problem are preliminary and we are doing further investigations on convergence properties of the algorithm and on the characteristics of L_p approximations in this context.

Application of the new IRLS method to the design of 2D FIR filters has also given encouraging results. Here again, it is difficult to apply the Remez exchange algorithm directly to the multi-dimensional approximation problem. Application of the IRLS to this problem is currently being investigated.

We designed 5×5, 7×7, 9×9, 41×41, and 71×71 filters to specifications used in [link], [link], [link], [link]. Our preliminary observations from these examples indicate the new IRLS method is faster and/or gives lower Chebyshev errors than any of the other methods [link]. Values of K in the 1.1 to 1.2 range were required for convergence. As for the complex approximation problem, further research is being done on convergence properties of the algorithm and on the characteristics of L_p approximations in this context.

Section Conclusions

We have proposed applying the iterative reweighted least squared error approach to the FIR digital filter design problem. We have shown how a large number of existing methods can be cast as variations on one basic successive approximation algorithm called Iterative Reweighted Least Squares. From this formulation we were able to understand the convergence characteristics of all of them and see why Lawson's method has experimentally been found to have slow convergence.

We have created a new IRLS algorithm by combining an improved acceleration scheme with Fletcher's and Kahng's Newton type methods to give a very good design method with good initial and final convergence properties. It is a significant improvement over the Rice-Usow-Lawson method.

The main contribution of the paper was showing how to use these algorithms with different p in different frequency bands to give a filter with different pass and stopband characteristics, how to solve the constrained L_p problem, and how the approach is used in complex approximation and in 2D filter design.

Minimum Phase Design

Here we design optimal approximations that can be “lifted" to give a positive function that when viewed as a magnitude squared, can be factored to give a minimum phase optimal design. However, the factoring can be a problem for long filters.

Window Function Design of FIR Filters

One should not use Hamming, Hanning, Blackman, or Bartlet windows for the design of FIR filters. They are appropriate for segmenting long data strings into shorter blocks to minimize the effects of blocking, but they do not design filters with any control over the transition band and do not design filters that are optimal in any meaningful sense.

The Kaiser window does have the ability to control the transition band. It also gives a fairly good approximation to a least squares approximation modified to reduce the Gibbs effect. However, the design is also not optimal in any meaningful sense and does not allow individual control of the widths of multiple transition bands. The spline transition function method gives the same control as the Kaiser window but does have a criterion of optimality and does allow independent control over individual transition bands. No window method allows any separate weighting of the error in different bands.

Properties of IIR Filters

Digital filters with an Infinite-duration Impulse Response (IIR) have characteristics that make them useful in many applications. This section develops and discusses the properties and characteristics of these filters[link].

Because of the feedback necessary in an implementation, the Infinite Impulse Response (IIR) filter is also called a recursive filter or, sometimes, an autoregressive moving-average filter (ARMA). In contrast to the FIR filter with a polynomial transfer function, the IIR filter has a rational transfer function. The transfer function being a ratio of polynomials means it has finite poles as well as zeros, and the frequency-domain design problem becomes a rational-function approximation problem in contrast to the polynomial approximation for the FIR filter[link]. This gives considerably more flexibility and power, but brings with it certain problems in both design and implementation[link], [link], [link].

The defining relationship between the input and output variables for the IIR filter is given by
Equation:

$$y\left(n\right)=\sum_{k=1}^{N}a\left(k\right)y\left(n-k\right)+\sum_{m=0}^{M}b\left(m\right)x\left(n-m\right).$$

The second summation in [link] is exactly the same moving average of the present plus past M values of the input that occurs in the definition of the FIR filter. The difference arises from the first summation, which is a weighted sum of the previous N output values. This is the feedback or recursive part which causes the response to an impulse input theoretically to endure forever. The calculation of each output term y(n) from [link] requires $N+M+1$ multiplications and $N+M$ additions. There are other algorithms or structures for calculating $y(n)$ that may require more or less arithmetic.

In addition to the number of calculations required to calculate each output term being a measure of efficiency, the amount of storage for coefficients and intermediate calculations is important. DSP chips are designed to efficiently implement calculations such as [link] by having a single cycle operation that multiplies a variable by a constant and accumulates it. In parallel with that operation, it is simultaneously calculating the address of the next variable.

Just as in the case of the FIR filter, the output of an IIR filter can also be calculated by convolution.
Equation:

$$y\left(n\right)=\sum_{k=0}^{\infty}h\left(k\right)x\left(n-k\right)$$

In this case, the duration of the impulse response $h(n)$ is infinite and, therefore, the number of terms in [link] is infinite. The $N+M+1$ operations required in [link] are clearly preferable to the infinite number required by [link]. This gives a hint as to why the IIR filter is very efficient. The details will become clear as the characteristics of the IIR filter are developed in this section.

Frequency-Domain Formulation of IIR Filters

The transfer function of a filter is defined as the ratio $Y(z)/X(z)$, where $Y(z)$ and $X(z)$ are the z-transforms of the output $y(n)$ and input $x(n)$, respectively. It is also the z-transform of the impulse response. Using the definition of the z-transform in Equation 32 from Discrete-Time Signals, the transfer function of the IIR filter defined in [link] is
Equation:

$$H\left(z\right)=\sum_{n=0}^{\infty}h\left(n\right)z^{-n}$$

This transfer function is also the ratio of the z-transforms of the $a(n)$ and $b(n)$ terms.
Equation:

$$H\left(z\right)=\frac{\sum_{n=0}^{M}b\left(n\right)z^{-n}}{\sum_{n=0}^{N}a\left(n\right)z^{-n}}=\frac{B(z)}{A(z)}$$

The frequency response of the filter is found by setting $z=e^{j\omega}$, which gives [link] the form
Equation:

$$H\left(\omega\right)=\sum_{n=0}^{\infty}h\left(n\right)e^{-j\omega n}$$

It should be recalled that this form assumes a sampling rate of $T=1$. To simplify notation, $H(\omega)$ is used to denote the frequency response rather than $H\left(e^{j\omega}\right)$.

This frequency-response function is complex-valued and consists of a magnitude and phase. Even though the impulse response is a function of the discrete variable n, the frequency response is a function of the continuous-frequency variable ω and is periodic with period 2π.

Unlike the FIR filter case, exactly linear phase is impossible for the IIR filter. It has been shown that linear phase is equivalent to symmetry of the impulse response. This is clearly impossible for the IIR filter with an impulse response that is zero for $n<0$ and nonzero for n going to infinity.

The FIR linear-phase filter allowed removing the phase from the design process. The resulting problem was a real-valued approximation problem requiring the solution of linear equations. The IIR filter design problem is more complicated. Linear phase is not possible, and the equations to be solved are generally nonlinear. The most common technique is to approximate the magnitude of the transfer function and let the phase take

care of itself. If the phase is important, it becomes part of the approximation problem, which then is often difficult to solve.

Calculation of the IIR Filter Frequency Response

As shown in another module, L equally spaced samples of $H(\omega)$ can be approximately calculated by taking an L-length DFT of $h(n)$ given in [link]. However, unlike for the FIR filter, this requires that the infinitely long impulse response be truncated to at least length-L. A more satisfactory alternative is to use the DFT to evaluate the numerator and denominator of [link] separately rather than to approximately evaluate [link]. This is accomplished by appending $L - N$ zeros to the $a(n)$ and $L - M$ zeros to the $b(n)$ from [link], and taking length-L DFTs of both to give
Equation:

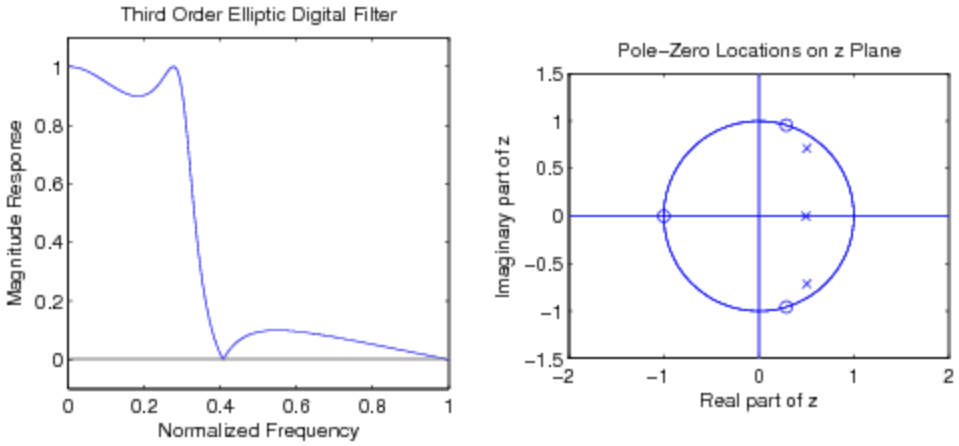
$$H\left(2\pi k/L\right) = \frac{\mathscr{DFT}\{b(n)\}}{\mathscr{DFT}\{a(n)\}}$$

where the division is a term-wise division of each of the L values of the DFTs as a function of k. This direct method of calculation is a straightforward and flexible technique that does not involve truncation of $h(n)$ and the resulting error. Even nonuniform spacing of the frequency samples can be achieved by altering the DFT as was suggested for the FIR filter. Because IIR filters are generally lower in order than FIR filters, direct use of the DFT is usually efficient enough and use of the FFT is not necessary. Since the $a(n)$ and $b(n)$ do not generally have the symmetries of the FIR $h(n)$, the DFTs cannot be made real and, therefore, the shifting and stretching techniques of other modules are not applicable.

As an example, the frequency-response plot of a third-order elliptic-function lowpass filter with a transfer function of
Equation:

$$H\left(z\right) = \frac{0.1335z^3 + 0.056z^2 + 0.056z + 0.1335}{z^3 - 1.507z^2 + 1.2646z - 0.3786}$$

is given in [link]a. The details for designing this filter are discussed in elsewhere. A similar performance for the magnitude response would require a length of 18 for a linear-phase FIR filter.



Frequency Response and Pole-Zero Locations of a Third-Order IIR Filter

Pole-Zero Locations for IIR Filters

The possible locations of the zeros of the transfer function of an FIR linear-phase filter were analyzed elsewhere. For the IIR filter, there are poles as well as zeros. For most applications, the coefficients $a(n)$ and $b(n)$ are real and, therefore, the poles and zeros occur in complex conjugate pairs or are real. A filter is stable if for any bounded input, the output is bounded. This implies the poles of the transfer function must be strictly inside the unit circle of the complex z plane. Indeed, the possibility of an unstable filter is a serious problem in IIR filter design, which does not exist for FIR filters. An important characteristic of any design procedure is the guarantee of stable designs, and an important ability in the analysis of a given filter is the determination of stability. For a linear filter analysis, this involves the zeros of the denominator polynomial of [link]. The location of the zeros of the

numerator, which are the zeros of $H(z)$, are important to the performance of the filter, but have no effect on stability.

If both the poles and zeros of a transfer function are all inside or on the unit circle of the z-plane, the filter is called minimum phase. The effects of a pole or zero at a radius of r from the origin of the z-plane on the magnitude of the transfer function are exactly the same as one at the same angle but at a radius of $1/r$. However, the effect on the phase characteristics is different. Since only stable filters are generally used in practice, all the poles must be inside the unit circle. For a given magnitude response, there are two possible locations for each zero that is not on the unit circle. The location that is inside gives the least phase shift, hence the name “minimum- phase" filter.

The locations of the poles and zeros of the example in [link] are given in [link]b.

Since evaluating the frequency response of a transfer function is the same as evaluating $H(z)$ around the unit circle in the z-plane, a comparison of the frequency-response plot in [link]a and the pole-zero locations in [link]b gives insight into the effects of pole and zero location on the frequency response. In the case where it is desirable to reject certain bands of frequencies, zeros of the transfer function will be located on the unit circle at locations corresponding to those frequencies.

By having both poles and zeros to describe an IIR filter, much more can be done than in the FIR filter case where only zeros exist. Indeed, an FIR filter is a special case of an IIR filter with a zero-order denominator. This generality and flexibility does not come without a price. The poles are more difficult to realize than the zeros, and the design is more complicated.

Summary

This section has given the basic definition of the IIR or recursive digital filter and shown it to a generalization of the FIR filter described in the previous chapters. The feedback terms in the IIR filter cause the transfer function to be a rational function with poles as well as zeros. This feedback

and the resulting poles of the transfer function give a more versatile filter requiring fewer coefficients to be stored and less arithmetic. Unfortunately, it also destroys the possibility of linear phase and introduces the possibility of instability and greater sensitivity to the effects of quantization. The design methods, which are more complicated than for the FIR filter, are discussed in another section, and the implementation, which also is more complicated, is discussed in still another section.

Design of Infinite Impulse Response (IIR) Filters by Frequency Transformations

The design of a digital filter is usually specified in terms of the characteristics of the signals to be passed through the filter. In many cases, the signals are described in terms of their frequency content. For example, even though it cannot be predicted just what a person may say, it can be predicted that the speech will have frequency content between 300 and 4000 Hz. Therefore, a filter can be designed to pass speech without knowing what the speech is. This is true of many signals and of many types of noise or interference. For these reasons among others, specifications for filters are generally given in terms of the frequency response of the filter.

The basic IIR filter design process is similar to that for the FIR problem:

1. Choose a desired response, usually in the frequency domain;
2. Choose an allowed class of filters, in this case, the Nth-order IIR filters;
3. Establish a measure of distance between the desired response and the actual response of a member of the allowed class; and
4. Develop a method to find the best allowed filter as measured by being closest to the desired response.

This section develops several practical methods for IIR filter design. A very important set of methods is based on converting Butterworth, Chebyshev I and II, and elliptic-function analog filter designs to digital filter designs by both the impulse- invariant method and the bilinear transformation. The characteristics of these four approximations are based on combinations of a Taylor's series and a Chebyshev approximation in the pass and stopbands. Many results from this chapter can be used for analog filter design as well as for digital design.

Extensions of the frequency-sampling and least-squared-error design for the FIR filter are developed for the IIR filter. Several direct iterative numerical methods for optimal approximation are described in this chapter. Prony's method and direct numerical methods are presented for designing IIR filters according to time-domain specifications.

The discussion of the four classical lowpass filter design methods is arranged so that each method has a section on properties and a section on design procedures. There are also design programs in the appendix. An experienced person can simply use the design programs. A less experienced designer should read the design procedure material, and a person who wants to understand the theory in order to modify the programs, develop new programs, or better understand the given ones, should study the properties section and consult the references.

Rational Function Approximation

The mathematical problem inherent in the frequency-domain filter design problem is the approximation of a desired complex frequency-response function $H_d(z)$ by a rational transfer function $H(z)$ with an Mth-degree numerator and an Nth-degree denominator for values of the complex variable z along the unit circle of $z = e^{j\omega}$. This approximation is achieved by minimizing an error measure between $H(\omega)$ and $H_d(\omega)$.

For the digital filter design problem, the mathematics are complicated by the approximation being defined on the unit circle. In terms of z, frequency is a polar coordinate variable. It is often much easier and clearer to formulate the problem such that frequency is a rectangular coordinate variable, in the way it naturally occurs for analog filters using the Laplace complex variable s. A particular change of complex variable that converts the polar coordinate variable to a rectangular coordinate variable is the bilinear transformation[link], [link], [link], [link].
Equation:

$$z = -\frac{s+1}{s-1}$$

The details of the bilinear and alternative transformations are covered elsewhere. For the purposes of this section, it is sufficient to observe[link], [link] that the frequency response of a filter in terms of the new variable is found by evaluating $H(s)$ along the imaginary axis, i.e., for $s = j\omega$. This is exactly how the frequency response of analog filters is obtained.

There are two reasons that the approximation process is often formulated in terms of the square of the magnitude of the transfer function, rather than the real and/or imaginary parts of the complex transfer function or the magnitude of the transfer function. The first reason is that the squared-magnitude frequency- response function is an analytic, real-valued function of a real variable, and this considerably simplifies the problem of finding a “best" solution. The second reason is that effects of the signal or interference are often stated in terms of the energy or power that is proportional to the square of the magnitude of the signal or noise.

In order to move back and forth between the transfer function $F(s)$ and the squared-magnitude frequency response $|F(j\omega)|^2$, an intermediate function is defined. The analytic complex-valued function of the complex variable s is defined by

Equation:

$$FF(s) = F(s)F(-s)$$

which is related to the squared magnitude by

Equation:

$$FF(s)|_{s=j\omega} = |F(j\omega)|^2$$

If

Equation:

$$F(j\omega) = R(\omega) + jI(\omega)$$

then

Equation:

$$|F(j\omega)|^2 = R(\omega)^2 + I(\omega)^2$$

Equation:

$$= (R(\omega) + jI(\omega))(R(\omega) - jI(\omega))$$

Equation:

$$= F(s)F(-s)|_{s=j\omega}$$

In this context, the approximation is arrived at in terms of $F(j\omega)$, and the result is an analytic function $FF(s)$ with a factor $F(s)$, which is the desired filter transfer function in terms of the rectangular variable s. A comparable function can be defined in terms of the digital transfer function using the polar variable z by defining
Equation:

$$HH(z) = H(z)H(1/z)$$

which gives the magnitude-squared frequency response when evaluated around the unit circle, i.e., $z = e^{j\omega}$.

The next section develops four useful approximations using the continuous-time Laplace transform formulation in s. These will be transformed into digital transfer functions by techniques covered in another module. They can also be used directly for analog filter design.

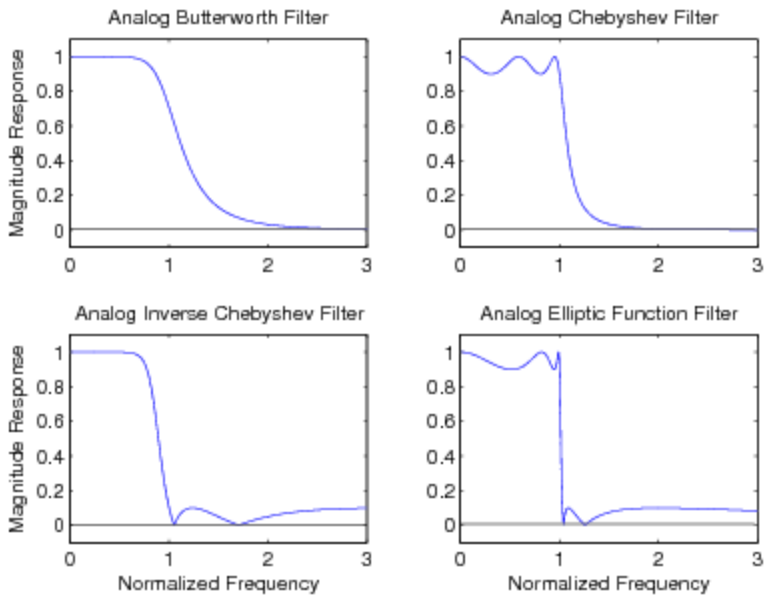
Classical Analog Lowpass Filter Approximations

Four basic filter approximations are considered to be standard. They are often developed and presented in terms of a normalized lowpass filter that can be modified to give other versions such as highpass or bandpass filters. These four forms use Taylor's series approximations and Chebyshev approximations in various combinations[link], [link], [link], [link]. It is interesting to note that none of these are defined in terms of a mean-squared error measure. Although it would be an interesting error criterion, the reason is that there is no closed-form solution to the LS-error approximation problem which is nonlinear for the IIR filter.

This section develops the four classical approximations in terms of the Laplace transform variable s. They can be used as prototype filters to be converted into digital filters or used directly for analog filter design.

The desired lowpass filter frequency response is similar to the case for the FIR filter. Here it is expressed in terms of the magnitude squared of the transfer function, which is a function of $s = j\omega$ and is illustrated in Figure 8 from FIR Digital Filters and Figure 1 from Least Squared Error Design of FIR Filters.

The Butterworth filter uses a Taylor's series approximation to the ideal at both $\omega = 0$ and $\omega = \infty$. The Chebyshev filter uses a Chebyshev (min-max) approximation across the passband and a Taylor's series at $\omega = \infty$. The Inverse or Type-II Chebyshev filter uses a Taylor's series approximation at $\omega = 0$ and a Chebyshev across the stopband. The elliptic-function filter uses a Chebyshev approximation across both the pass and stopbands. The squared- magnitude frequency response for these approximations to the ideal is given in [link], and the design is developed in the following sections.



Frequency Responses of the Four Classical

Lowpass IIR Filter Approximations

Butterworth Filter Properties

This section develops the properties of the Butterworth filter which has as its basic concept a Taylor's series approximation to the desired frequency response. The measure of the approximation is the number of terms in the Taylor's series expansion of the actual frequency response that can be made equal to those of the desired frequency response. The optimal or best solution will have the maximum number of terms equal. The Taylor's series is a power series expansion of a function in the form of

Equation:

$$F(\omega) = K_0 + K_1\omega + K_2\omega^2 + K_3\omega^3 + \cdots$$

where

Equation:

$$K_0 = F(0), \quad K_1 = \left.\frac{dF(\omega)}{d\omega}\right|_{\omega=0}, \quad K_2 = (1/2)\left.\frac{d^2F(\omega)}{d\omega^2}\right|_{\omega=0}, etc.,$$

with the coefficients of the Taylor's series being proportional to the various order derivatives of $F(\omega)$ evaluated at $\omega = 0$. A basic characteristic of this approach is that the approximation is all performed at one point, i.e., at one frequency. The ability of this approach to give good results over a range of frequencies depends on the analytic properties of the response.

The general form for the squared-magnitude response is an even function of ω and, therefore, is a function of ω^2 expressed as

Equation:

$$FF(j\omega) = \frac{d_0 + d_2\omega^2 + d_4\omega^4 + \ldots + d_{2M}\omega^{2M}}{c_0 + c_2\omega^2 + c_4\omega^4 + \ldots c_{2N}\omega^{2N}}$$

In order to obtain a solution that is a lowpass filter, the Taylor's series expansion is performed around $\omega = 0$, requiring that $FF(0) = 1$ and that $FF(j\infty) = 0$, (i.e., $d_0 = c_0$, $N > M$, and $c_{2N} \neq 0$). This is written as

Equation:

$$FF(j\omega) = 1 + E(\omega)$$

Combining [link] and [link] gives
Equation:

$$d_0 + d_2\omega^2 + \cdots + d_2Mw = c_0 + c_2w + \cdots + c_{2N}\omega^{2N} + E\left(\omega\right)\left[c_0 + c_2\omega + \cdots\right]$$

The best Taylor's approximation requires that $FF(j\omega)$ and the desired ideal response have as many terms as possible equal in their Taylor's series expansion at a given frequency. For a lowpass filter, the expansion is around $\omega = 0$, and this requires $E(\omega)$ have as few low-order ω terms as possible. This is achieved by setting
Equation:

$$c_0 = d_0, \quad c_2 = d_2, \quad \cdots \quad c_{2M} = d_{2M}, \cdots \quad c_{2M+2} = 0, \quad c_{2N-2} = 0, \quad c_{2N} \neq 0$$

Because the ideal response in the passband is a constant, the Taylor's series approximation is often called “maximally flat".

[link] states that the numerator of the transfer function may be chosen arbitrarily. Then by setting the denominator coefficients of FF(s) equal to the numerator coefficients plus one higher-order term, an optimal Taylor's series approximation is achieved [link].

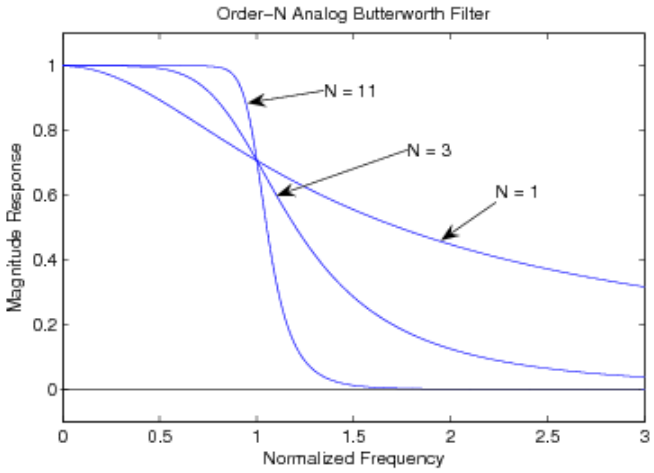
Since the numerator is arbitrary, its coefficients can be chosen for a Taylor's approximation to zero at $\omega = \infty$. This is accomplished by setting $d_0 = 1$ and all other d's equal zero. The resulting magnitude-squared function is[link]
Equation:

$$FF\left(j\omega\right) = \frac{1}{1 + c_{2N}\omega^{2N}}$$

The value of the constant c_{2N} determines at which value of ω the transition of passband to stopband occurs. For this development, it is normalized to $c_{2N} = 1$, which causes the transition to occur at $\omega = 1$. This gives the simple form for what is called the Butterworth filter
Equation:

$$FF\left(j\omega\right) = \frac{1}{1 + \omega^{2N}}$$

This approximation is sometimes called “maximally flat" at both $\omega = 0$ and $\omega = \infty$, since it is simultaneously a Taylor's series approximation to unity at $\omega = 0$ and to zero at $\omega = \infty$. A graph of the resulting frequency response function is shown in [link] for several N.



Frequency Responses of the Butterworth Lowpass Filter Approximation

The characteristics of the normalized Butterworth filter frequency response are:

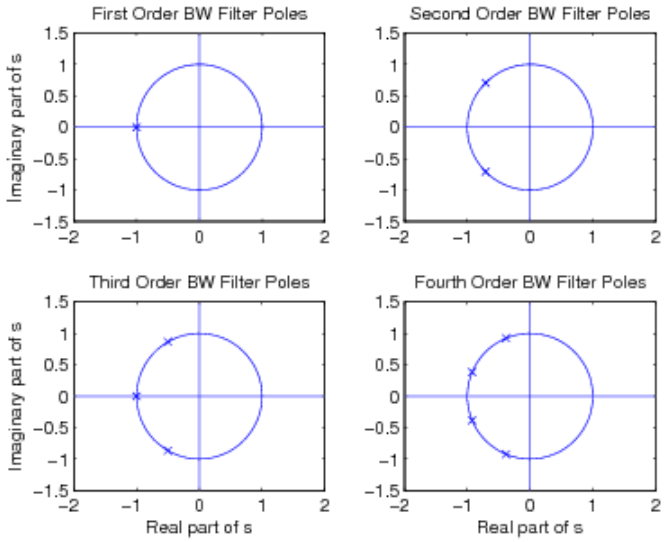
- Very close to the ideal near $\omega = 0$ and $\omega = \infty$,
- Very smooth at all frequencies with a monotonic decrease from $\omega = 0$ to ∞, and
- Largest difference between the ideal and actual responses near the transition at $\omega = 1$ where $|F(j1)|^2 = 1/2$.

Although not part of the approximation addressed, the phase curve is also very smooth.

An important feature of the Butterworth filter is the closed- form formula for the solution, $F(s)$. The expression for $FF(s)$ may be determined as
Equation:

$$F(s)F(-s) = \frac{1}{1 + (-s^2)^N}$$

This function has $2N$ poles evenly spaced around a unit radius circle and $2N$ zeros at infinity. The determination of $F(s)$ is very simple. In order to have a stable filter, $F(s)$ is selected to have the N left-hand plane poles and N zeros at infinity; $F(-s)$ will necessarily have the right-hand plane poles and the other N zeros at infinity. The location of these poles on the complex s plane for $N = 1, 2, 3,$ and 4 is shown in [link].



Pole Locations for Analog Butterworth Filter Transfer Function on the Complex s Plane

Because of the geometry of the pole positions, simple formulas are easy to derive for the pole locations. If the real and imaginary parts of the pole location are denoted as
Equation:

$$s = u + jw$$

the locations of the N poles are given by
Equation:

$$u_k = -\cos\left(k\pi/2N\right)$$

Equation:

$$\omega_k = \sin\left(k\pi/2N\right)$$

for N values of k where
Equation:

$$k = \pm 1, \pm 3, \pm 5, \ldots, \pm (N-1) \quad \text{for N even}$$

Equation:

$$k = 0, \pm 2, \pm 4, \ldots, \pm (N-1) \quad \text{for N odd}$$

Because the coefficients of the numerator and denominator polynomials of $F(s)$ are real, the roots occur in complex conjugate pairs. The conjugate pairs in [link],[link] can be combined to be the roots of second-order polynomials so that for N even, $F(s)$ has the partially factored form of
Equation:

$$F(s) = \prod_k \frac{1}{s^2 + 2\cos(k\pi/2N)s + 1}$$

for $k = 1, 3, 5, \ldots, N-1$. For N odd, $F(s)$ has a single real pole and, therefore, the form
Equation:

$$F(s) = \frac{1}{s+1} \prod_k \frac{1}{s^2 + 2\cos(k\pi/2N)s + 1}$$

for $k = 2, 4, 6, \cdots, N-1$

This is a convenient form for the cascade and parallel realizations discussed in elsewhere.

A single formula for the pole locations for both even and odd N is
Equation:

$$u_k = -\sin((2k+1)\pi/2N)$$

Equation:

$$\omega_k = \cos((2k+1)\pi/2N)$$

for N values of k where $k = 0, 1, 2, \ldots, N-1$

One of the important features of the Butterworth filter design formulas is that the pole locations are found by independent calculations which do not depend on each other or on factoring a polynomial. A FORTRAN program which calculates these values is given in the appendix as Program 8. Mathworks has a powerful command for designing analog and digital Butterworth filters.

The classical form of the Butterworth filter given in [link] is discussed in many books [link], [link], [link], [link], [link]. The less well-known form given in [link] also has many useful applications [link]. If the frequency location of unwanted signals is known, the zeros of the transfer function given by the numerator can be set to best reject them. It is then possible to choose the pole locations so as to have a passband as flat as the classical Butterworth filter by using [link]. Unfortunately, there are no formulas for the pole locations; therefore, the denominator polynomial must be factored.

Summary
This section has derived design procedures and formulas for a class of filter transfer functions that approximate the ideal desired frequency response by a Taylor's series. If the approximation is made at $\omega = 0$ and $\omega = \infty$, the resulting filter is called a Butterworth filter and the response is called maximally-flat at zero and infinity. This filter has a very smooth frequency response and, although not explicitly designed for, has a smooth phase response. Simple formulas for the pole locations were derived and are implemented in the design program in the appendix of this book.

Butterworth Filter Design Procedures

This section considers the process of going from given specifications to use of the approximation results derived in the previous section. The Butterworth filter is the simplest of the four classical filters in that all the approximation effort is placed at two frequencies: $\omega = 0$ and $\omega = \infty$. The transition from passband to stopband occurs at a normalized frequency of $\omega = 1$. Assuming that this transition frequency or bandedge can later be scaled to any desired frequency, the only parameter to be chosen in the design process is the order N.

The filter specifications that are consistent with what is optimized in the Butterworth filter are the degree of “flatness" at $\omega = 0$ (DC) and at $\omega = \infty$. The higher the order, the flatter the frequency response at these two points. Because of the analytic nature of rational functions, the flatter the response is at $\omega = 0$ and $\omega = \infty$, the closer it stays to the desired response throughout the whole passband and stopband. An

indirect consequence of the filter order is the slope of the response at the transition between pass and stopband. The slope of the squared-magnitude frequency response at $\omega = 1$ is
Equation:

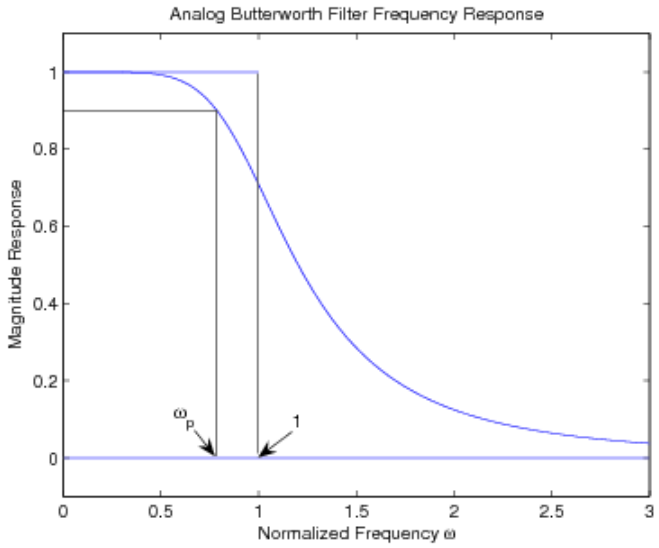
$$s = FF'(j1) = -N/2$$

The effects of the increased flatness and increased transition slope of the frequency response as N increases are illustrated in Figure 1 from Design of Infinite Impulse Response (IIR) Filters by Frequency Transformations.

In some cases specifications state the response must stay above or below a certain value over a given frequency band. Although this type of specification is more compatible with a Chebyshev error optimization, it is possible to design a Butterworth filter to meet the requirements. If the magnitude of the frequency response of the filter over the passband of $0 < \omega < \omega_P$ must remain between unity and G, where $\omega_p < 1$ and $G < 1$, the required order is found by the smallest integer N satisfying
Equation:

$$N \geq \frac{\log\left((1/G)^2 - 1\right)}{1\log\left(\omega_p\right)}$$

This is illustrated in [link] where $|F|$ must remain above 0.9 for ω up to 0.9, i.e., G = 0.9 and ω_p = 0.9. These requirements require an order of at least $N = 7$.



Passband Specifications for Designing a Butterworth Filter

If stopband performance is stated in the form of requiring that the response stay below a certain value for frequency above a certain value, i.e., $|F| < G$ for $\omega > \omega_s$, the order is determined by the same formula [link] with ω_p replaced by ω_s.

Note $|F(j1)| = 1/\sqrt{(2)}$ which is called the “half power" frequency because $|F(j1)|^2 = 1/2$. This frequency is normalized to one for the theory but can be scaled to any value for applications.

Example:
Design of a Butterworth Lowpass IIR Filter
To illustrate the calculations, a lowpass Butterworth filter is designed. It is desired that the frequency response stay above 0.8 for frequencies up to 0.9. The formula [link] for determining the order gives a value of 2.73; therefore, the order is three. The analytic function corresponding to the squared-magnitude frequency response in [link] is
Equation:

$$|F(j\omega)|^2 = \frac{1}{1+\omega^6}$$

The transfer function corresponding to the left-half-plane poles of F'(s) are calculated from [link] to give

Equation:

$$F(s) = \frac{1}{(s+1)(s+0.5+j0.866)(s+0.5-j0.866)}$$

Equation:

$$F(s) = \frac{1}{(s+1)(s^2+s+1)}$$

Equation:

$$F(s) = \frac{1}{s^3+2s^2+2s+1}$$

The frequency response is obtained by setting $s = j\omega$ which has a plot illustrated in [link] for $N = 3$. The pole locations are the same as shown in [link]c.

Chebyshev Filter Properties

Chebyshev Filter Properties

The Butterworth filter does not give a sufficiently good approximation across the complete passband in many cases. The Taylor's series approximation is often not suited to the way specifications are given for filters. An alternate error measure is the maximum of the absolute value of the difference between the actual filter response and the ideal. This is considered over the total passband. This is the Chebyshev error measure and was defined and applied to the FIR filter design problem. For the IIR filter, the Chebyshev error is minimized over the passband and a Taylor's series approximation at $\omega = \infty$ is used to determine the stopband performance. This mixture of methods in the IIR case is called the Chebyshev filter, and simple design formulas result, just as for the Butterworth filter.

The design of Chebyshev filters is particularly interesting, because the results of a very elegant theory insure that constructing a frequency-response function with the proper form of equal ripple in the error will result in a minimum Chebyshev error without explicitly minimizing anything. This allows a straightforward set of design formulas to be derived which can be viewed as a generalization of the Butterworth formulas [link], [link].

The form for the magnitude squared of the frequency-response function for the Chebyshev filter is
Equation:

$$\left|F\left(j\omega\right)\right|^2 = \frac{1}{1+\epsilon^2 C_N(\omega)^2}$$

where $C_N\left(\omega\right)$ is an Nth-order Chebyshev polynomial and ϵ is a parameter that controls the ripple size. This polynomial in ω has very special characteristics that result in the optimality of the response function [link].

CHEBYSHEV POLYNOMIALS

The Chebyshev polynomial is a powerful function in approximation theory. Although the function is a polynomial, it is best defined and developed in terms

of trigonometric functions by[link], [link], [link], [link].
Equation:

$$C_N\left(\omega\right) = \cos\left(N\,\cos^{-1}\left(\omega\right)\right)$$

where $C_N\left(\omega\right)$ is an Nth-order, real-valued function of the real variable ω. The development is made clearer by introducing an intermediate complex variable φ.
Equation:

$$C_N\left(\omega\right) = \cos\left(N\varphi\right)$$

where
Equation:

$$\omega = \cos\left(\varphi\right)$$

Although this definition of $C_N\left(\omega\right)$ may not at first appear to result in a polynomial, the following recursive relation derived from [link] shows that it is a polynomial.
Equation:

$$C_{N+1}\left(\omega\right) = 2\omega C_N\left(\omega\right) - C_{N-1}\left(\omega\right)$$

From [link], it is clear that $C_0 = 1$ and $C_1 = \omega$, and from [link], it follows that
Equation:

$$C_2 = 2\omega^2 - 1$$

Equation:

$$C_3 = 4\omega^3 - 3\omega$$

Equation:

$$C_4 = 8\omega^4 - 8\omega^2 + 1$$

etc.

Other relations useful for developing these polynomials are
Equation:

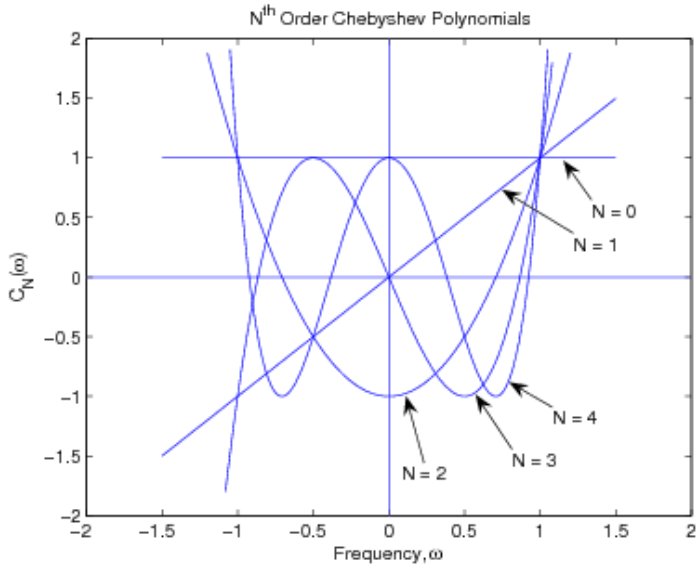
$$C_N^2(\omega) = (C_{2N}(\omega) + 1)/2$$

Equation:

$$C_{MN}(\omega) = C_M(C_N(\omega))$$

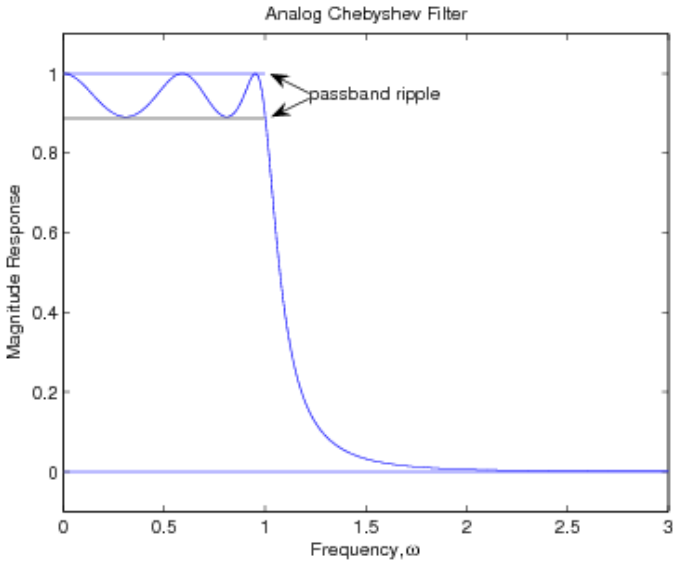
where M and N are coprime.

These are remarkable functions [link]. They oscillate between +1 and -1 for $-1 < \omega < 1$ and go monotonically to +/- infinity outside that domain. All N of their zeros are real and fall in the domain of $-1 < \omega < 1$, i.e., C_N is an equal ripple approximation to zero over the range of ω from -1 to +1. In addition, the values for ω where C_N reaches its local maxima and minima and is zero are easily calculated from [link] and [link]. For $-1 < \omega < 1$, a plot of $C_N(\omega)$ can be made using the concept of Lissajous figures. Example plots for C_0, C_1, C_2, C_3, and C_4 are shown in [link].



Chebyshev Polynomials for N = 0, 1, 2, 3, and 4

The filter frequency-response function for $N = 5$ is given in [link] showing the passband ripple in terms of the parameter ϵ.



Fifth Order Chebyshev Filter Frequency Response

The approximation parameters must be clearly understood. The passband ripple is defined to be the difference between the maximum and the minumum of $|F|$ over the passband frequencies of $0 < \omega < 1$. There can be confusion over this point as two definitions appear in the literature. Most digital [link], [link], [link], [link] and analog [link] filter design books use the definition just stated. Approximation literature, especially concerning FIR filters, use one half this value which is a measure of the maximum error, $||F|-|F_d||$, where $|F_d|$ is the center line in the passband of [link], which $|F|$ oscillates around.

The Chebyshev theory states that the maximum error over that band is minimum and that this optimal approximation function has equal ripple over the pass band. It is easy to see that e in [link] determines the ripple in the passband and the order N determines the rate that the response goes to zero as ω goes to infinity.

Pole Locations

A method for finding the pole locations for the Chebyshev filter transfer function is next developed. The details of this section can be skipped and the results in [link],[link] used if desired.

From [link], it is seen that the poles of $FF(s)$ occur when
Equation:

$$1+\epsilon^2 C_N^2\left(s/j\right)=0$$

or
Equation:

$$C_N=\pm\frac{j}{\epsilon}$$

From [link], define $\varphi=\cos^{-1}\left(\omega\right)$ with real and imaginary parts given by
Equation:

$$\varphi=\cos^{-1}\left(\omega\right)=u+jv$$

This gives,
Equation:

$$C_N=\cos\left(N\varphi\right)=\cos\left(Nu\right)\cosh\left(N\nu\right)-j\sin\left(Nu\right)\sinh\left(N\nu\right)=\pm\frac{j}{\epsilon}$$

which implies the real part of C_N is zero. This requires
Equation:

$$\cos\left(Nu\right)\cosh\left(N\nu\right)=0$$

which implies
Equation:

$$\cos\left(Nu\right)=0$$

which in turn implies that u takes on values of
Equation:

$$u = u_k = (2k+1)\pi/2N, \qquad k = 0, 1, \ldots N-1$$

For these values of u, $\sin\,(nu) = \pm 1$, we have
Equation:

$$\sinh\,(N\nu) = 1/\epsilon$$

which requires ν to take on a value of
Equation:

$$\nu = \nu_0 = \left(\sinh^{-1}\,(1/\epsilon)\right)/N$$

Using $s = j\omega$ gives
Equation:

$$s = j\omega = j\,\cos\,(\varphi) = j\,\cos\,(u + j\nu) = j\,\cos\,((2k+1)\pi/2N + j\nu_0)$$

which gives the location of the N poles in the s plane as
Equation:

$$s_k = \sigma_k + j\omega_k$$

where
Equation:

$$\sigma_k = -\,\sinh\,(\nu_0)\,\cos\,(k\pi/2N)$$

Equation:

$$\omega_k = \cosh\,(\nu_0)\,\sin\,(k\pi/2N)$$

for N values of k where
Equation:

$$k = \pm 1, \pm 3, \pm 5, \cdots, \pm (N-1) \quad \text{for N even}$$

Equation:

$$k = 0, \pm 2, \pm 4, \cdots, \pm (N-1) \quad \text{for N odd}$$

A partially factored form for F(s) can be derived using the same approach as for the Butterworth filter. For N even, the form is
Equation:

$$F\left(s\right) = \prod_k \frac{1}{s^2 - 2\sigma_k s + \left(\sigma_k^2 + \omega_k^2\right)}$$

for $k = 1, 3, 5, \cdots, N-1$. For N odd, $F(s)$ has a single real pole and, therefore, the form
Equation:

$$F\left(s\right) = \frac{1}{\sinh\left(\nu_0\right)} F\left(s\right) = \prod_k \frac{1}{s^2 - 2\sigma_k s + \left(\sigma_k^2 + \omega_k^2\right)}$$

for $k = 2, 4, 6, \ldots, N-1$ This is a convenient form for the cascade and parallel realizations.

A single formula for both even and odd N is
Equation:

$$\sigma = -\sinh\left(\nu_0\right) \sin\left((2k+1)\pi/2N\right)$$

Equation:

$$\omega_k = \cosh\left(\nu_0\right) \cos\left((2k+1)\pi/2N\right)$$

for N values of k where $k = 0, 1, 2, \cdots, N-1$

Note the similarity to the pole locations for the Butterworth filter. Cross multiplying, squaring, and adding the terms in [link],[link] gives
Equation:

$$\left(\frac{\sigma_k}{\sinh\left(\nu_0\right)}\right)^2+\left(\frac{\omega_k}{\cosh\left(\nu_0\right)}\right)^2=1$$

This is the equation for an ellipse and shows that the poles of a Chebyshev filter lie on an ellipse similar to the way the poles of a Butterworth filter lie on a circle[link], [link], [link], [link], [link], [link].

Summary

This section has developed the classical Chebyshev filter approximation which minimizes the maximum error over the passband and uses a Taylor's series approximation at infinity. This results in the error being equal ripple in the passband. The transfer function was developed in terms of the Chebyshev polynomial and explicit formulas were derived for the location of the transfer function poles. These can be expressed as a modification of the pole locations for the Butterworth filter and are implemented in the appendix.

It is possible to develop a theory for Chebyshev passband approximation and arbitrary zero location similar to the Taylor's series result in Equation 5 from Butterworth Filter Properties.

Chebyshev Filter Design Procedures

The Chebyshev filter has a passband optimized to minimize the maximum error over the complete passband frequency range, and a stopband controlled by the frequency response being maximally flat at $\omega = \infty$. The passband ripple and the filter order are the two parameters to be determined by the specifications.

The form for the specifications that is most consistent with the Chebyshev filter formulation is a maximum allowed error in the passband and a desired degree of “flatness" at $\omega = \infty$. The slope of the response near the transition from pass to stopband at $\omega = 1$ becomes steeper as both the order increases and the allowed passband error ripple increases. The dropoff is more rapid than for the Butterworth filter[link].

As stated earlier, the design parameters must be clearly understood to obtain a desired result. The passband ripple is defined to be the difference between the maximum and the minimum of $|F|$ over the passband frequencies of $0 < \omega < 1$. There can be confusion over this point as two definitions appear in the literature. Most digital [link], [link], [link] and analog [link] filter design books use the definition just stated. Approximation literature, especially concerning FIR filters, use half this value which is a measure of the maximum error, $||F|-|F_d||$, where $|F_d|$ is the center line in the passband around which $|F|$ oscillates. The following formulas relate the passband ripple δ, the passband ripple a in positive dB, and the transfer function parameter ϵ.
Equation:

$$a = 10 \log \left(1 + \epsilon^2\right) = -20 \log \left(1 - \delta\right),$$

Equation:

$$\epsilon = \sqrt{\frac{2\delta - \delta^2}{1 - 2\delta + \delta^2}} = \sqrt{10^{a/10} - 1},$$

Equation:

$$\delta = 1 - 10^{-a/20} = 1 - \frac{1}{\sqrt{1 + \epsilon^2}}$$

In some cases, stopband performance is not given in terms of degree of flatness at $\omega = \infty$, but in terms of a maximum allowed magnitude G in the stopband above a certain frequency ω_s, i.e., $G > |F| > 0$ for $1 < \omega_s < \omega < \infty$. For a given ϵ, this will determine the order as the smallest positive integer satisfying
Equation:

$$N \geq \frac{\cosh^{-1}\left(\frac{\sqrt{1-G^2}}{\epsilon G^2}\right)}{\cosh^{-1}\left(\omega_s\right)}$$

The design of a Chebyshev filter involves the following steps:

- The maximum-allowed passband variation must be in the form of δ or a. From this, the parameter ϵ is calculated using [link].

- The order N is determined by the desired flatness at $\omega = \infty$ or a maximum-allowed response for frequencies above ω_s using [link].
- ν_0 is calculated from ϵ and n using [link], and the scale factors $\sinh(\nu_0)$ and $\cosh(\nu_0)$ are then determined.
- The pole locations are calculated from [link] or [link]. This can be done by scaling the poles of a Butterworth prototype filter.
- These pole locations are combined in [link] and [link] to give the final filter transfer function.

This process is easily programmed for computer aided design as illustrated in Program 8 in the appendix.

If the design procedure uses [link] to determine the order and the right-hand side of the equation is not exactly an integer, it is possible to improve on the specifications. Direct use of the order with ϵ from [link] gives a stopband gain at ω_s that is less than G, or the same design can be viewed as giving the maximum-allowed gain G at a lower frequency than ω_s. An alternate approach is to solve [link] for a new value of ϵ, then cause [link] to be an equation with the specified ω_s and G. This gives a filter that exactly meets the stopband specifications and gives a smaller passband ripple than originally requested. A similar set of alternatives exists for the elliptic-function filter.

Example:
The Design of a Chebyshev Lowpass Filter.
The design specifications require a maximum passband ripple of $\delta = 0.1$ or $a = 0.91515$ dB, and can allow no greater response than $G = 0.2$ for frequencies above $\omega_s = 1.6$ radians per second.
Given $\delta = 0.1$ or $a = 0.91515$, equation [link] implies
Equation:

$$\epsilon = 0.484322$$

Given $G = 0.2$ and $\omega_s = 1.6$, equation [link] implies an order of $N = 3$. From ϵ and N, ν_0 is 0.49074 from [link] and
Equation:

$$\sinh(\nu_0) = 0.510675$$

Equation:

$$\cosh(\nu_0) = 1.122849$$

These multipliers are used to scale the root locations of the example third-order Butterworth filter to give

Equation:

$$F(s) = \frac{1}{(s + 0.51067)(s + 0.25534 + j0.97242)(s + 0.25534 - j0.97242)}$$

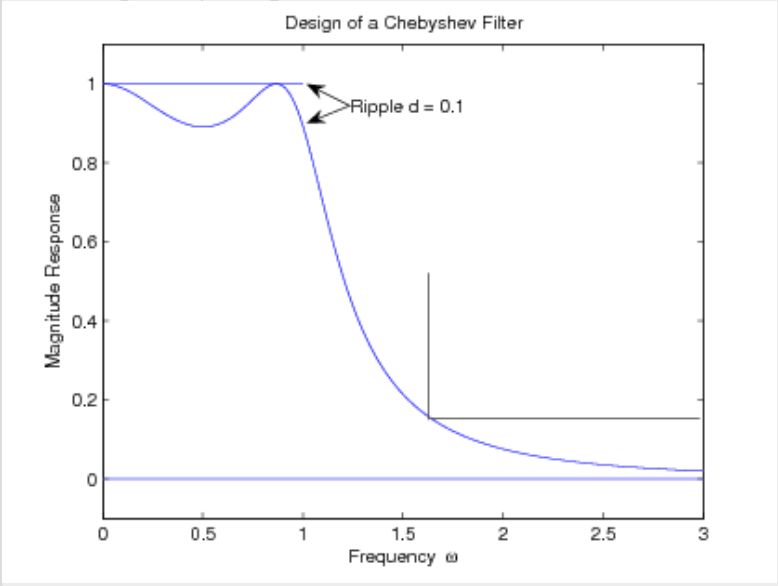
Equation:

$$F(s) = \frac{1}{(s + 0.51067)\left(s^2 + 0.510675s + 1.010789\right)}$$

Equation:

$$F(s) = \frac{1}{s^3 + 102135s^2 + 1.271579s + 0.516185}$$

The frequency response is shown in [link]

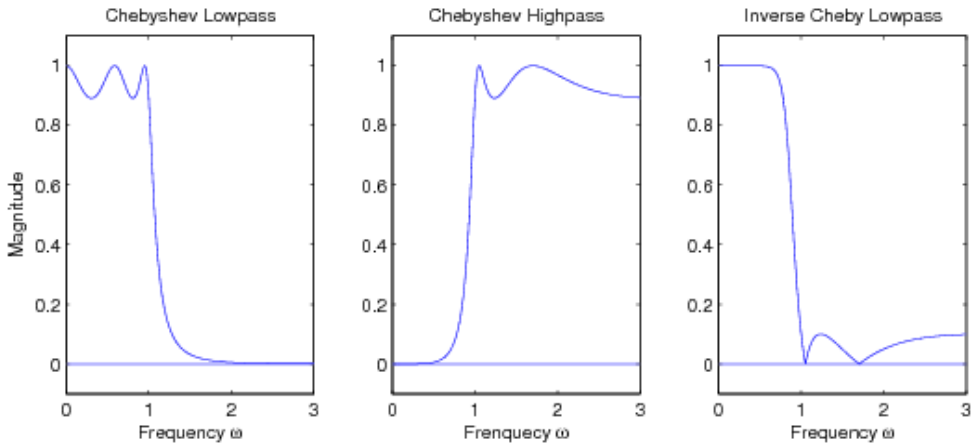


Example Design of a Third Order Chebyshev Filter Frequency Response

Inverse-Chebyshev Filter Properties

A second form of the mixture of a Chebyshev approximation and a Taylor's series approximation is called the Inverse Chebyshev filter or the Chebyshev II filter. This error measure uses a Taylor's series for the passband just as for the Butterworth filter and minimizes the maximum error over the total stopband. It reverses the types of approximation used in the preceding section. A fifth-order example is illustrated in Figure 1c from Design of Infinite Impulse Response (IIR) Filters by Frequency Transformations and [link]c.

Rather than developing the approximation directly, it is easier to modify the results from the regular Chebyshev filter. First, the frequency variable ω in the regular Chebyshev filter, described in [link], is replaced by $1/\omega$, which interchanges the characteristics at ω equals zero and infinity and does not change the performance at ω equals unity. This converts a Chebyshev lowpass filter into a Chebyshev highpass filter as illustrated in [link] moving from the first to second frequency response.



Lowpass Chebyshev to Highpass Chebyshev to Lowpass Inverse Chebyshev

This highpass characteristic is subtracted from unity to give the desired lowpass inverse-Chebyshev frequency response illustrated in [link]c. The resulting magnitude-squared frequency- response function is given by

Equation:

$$FF\left(j\omega\right)=\frac{\epsilon^2C_N^2\left(1/\omega\right)}{1+\epsilon^2C_N^2\left(1/\omega\right)}$$

Zero Locations

The zeros of the Chebyshev polynomial $C_N\left(\omega\right)$ are easily found by
Equation:

$$C_N\left(\omega\right)=0\Rightarrow N\cos^{-1}\left(\omega\right)=(2k+1)\pi/2$$

which requires
Equation:

$$\omega_k=\cos\left((2k+1)\pi/2N\right)$$

for $k=0,1,\cdots N-1$, or
Equation:

$$\omega_k=\sin\left(k\pi/2N\right)$$

for $k=0,\pm2,\pm4,\ldots,\pm(N-1)$: N odd

$k=\pm1,\pm3,\pm5,\ldots,\pm(N-1)$: N even

The zeros of the inverse-Chebyshev filter transfer function are derived from [link] and [link] to give
Equation:

$$\omega_{zk}=1/\left(\cos\left((2k+1)\pi/2N\right)\right)$$

The zero locations are not a function of ϵ, i.e., they are independent of the stopband ripple.

Pole Locations

The pole locations are the reciprocal of those for the regular Chebyshev filter. If the poles for the inverse filter are denoted by
Equation:

$$s'_k = \sigma'_k + j\omega'_k$$

the locations are
Equation:

$$\sigma'_k = \frac{\sigma_k}{\sigma_k^2 + \omega_k^2}$$

Equation:

$$\omega'_k = \frac{\omega_k}{\sigma_k^2 + \omega_k^2}$$

Although this gives a straightforward formula for calculating the location of the poles and zeros of the inverse- Chebyshev filter, they do not lie on a simple geometric curve as did those for the Butterworth and Chebyshev filters. Note that the conditions for a Taylor's series approximation with preset zero locations are satisfied.

A partially factored form for the Butterworth filter and for the Chebyshev filter can be written for the inverse-Chebyshev filter using the zero locations from [link] and the pole locations from the regular Chebyshev filter. For N even, this becomes
Equation:

$$F\left(s\right) = \frac{\prod_k \left(s^2 + \omega_{zk}^2\right)}{\prod_k \left(s^2 - 2\left(\sigma_k / \left(\sigma_k^2 + \omega_k^2\right)\right)s + 1 / \left(\sigma_k^2 + \omega_k^2\right)\right.}$$

for $k = 1, 3, 5, \cdots, N-1$. For N odd, F(s) has a single pole, and therefore, is of the form
Equation:

$$F(s) = \frac{\prod_k \left(s^2 + \omega_{zk}^2\right)}{\left(s + 1/\sinh\left(\nu_0\right)\right) \prod_k \left(s^2 - 2\left(\sigma_k / \left(\sigma_k^2 + \omega_k^2\right)\right) s + 1/\left(\sigma_k^2 + \omega_k^2\right)\right.}$$

for $k = 2, 4, 6, \cdots, N-1$

Because of the relationships between the locations of the poles of the Butterworth, Chebyshev, and inverse-Chebyshev filters, it is easy to write a design program with many common calculations. That is illustrated in the program in the appendix.

Inverse-Chebyshev Filter Design Procedures

The natural form for the specifications of an inverse-Chebyshev filter is in terms of the flatness of the response at ω to determine the passband, and a maximum allowable response in the stopband. The filter order and the stopband ripple are the parameters to be determined by the specifications. The rate of dropoff near the transition from pass to stopband is similar to the regular Chebyshev filter. Because practical specifications often allow more passband ripple than stopband ripple, the regular Chebyshev filter will usually have a sharper dropoff than the inverse-Chebyshev filter. Under those conditions, the inverse-Chebyshev filter will have a smoother phase response and less time-domain echo effects.

The stopband ripple d is simply defined as the maximum value that $|F(j\omega)|$ assumes in the stopband, which is the set of frequencies $1 < \omega < \infty$. An alternative specification is the minimum-allowed attenuation over stopband expressed in dB as b. The following formulas relate the stopband ripple δ, the stopband attenuation b in positive dB, and the transfer function parameter ϵ in [link]

Equation:

$$\epsilon = \frac{\delta}{\sqrt{1 - \delta^2}}$$

Equation:

$$\delta = \frac{\epsilon}{\sqrt{1 + \epsilon^2}}$$

Equation:

$$b = -10 \log \left(\epsilon^2 / \left(1 + \epsilon^2\right)\right) = -20 \log \left(d\right)$$

In some cases passband performance is not given in terms of degree of flatness at $\omega = 0$, but in terms of a minimum-allowed magnitude G in the passband up to a certain frequency ω_p, i.e., $1 > |F| > G$ for $0 < \omega < \omega_p < 1$. For a given ϵ, this requirement will determine the order as the smallest positive integer satisfying
Equation:

$$N > \frac{\cosh^{-1}\left(G / \left(\epsilon \sqrt{1 - G^2}\right)\right)}{\cosh^{-1}\left(1/\omega_p\right)}$$

The design of an inverse-Chebyshev filter is summarized in the following steps:

1. The maximum-allowed stopband response must be given in the form of δ or b. From this, the parameter ϵ is calculated using [link].
2. The order N is determined from the desired flatness at $\omega = 0$, or from a minimum allowed response for frequencies up to ω_p using [link].
3. ν_0 and $\sinh\left(\nu_0\right)$ and $\cosh\left(\nu_0\right)$ are calculated just as for the regular Chebyshev filter.
4. The pole locations for the prototype Chebyshev filter are calculated from [link] and [link] and then "inverted" according to [link] to give the inverse-Chebyshev filter pole locations.
5. The pole locations are combined in [link] to give the final filter transfer function denominator.
6. The zero locations are calculated from [link] and combined with the pole locations to give the total transfer function [link] or [link].

Example:
Example Design of an Inverse-Chebyshev Filter
A third-order inverse-Chebyshev lowpass filter is desired with a maximum-allowed stopband ripple of $d = 0.1$ or $b = 20$ dB. This corresponds to an ϵ of 0.100504 and, together with $N = 3$, results in a $\nu_0 = 0.99774$. The scale factors are sinh$= 1.171717$ and cosh$= 1.540429$. The prototype Chebyshev filter transfer function is

Equation:

$$F(s) = \frac{1}{(s + 1.1717)\left(s^2 + 1.1717s + 2.0404\right)}$$

The zeros are calculated from [link], and the poles of the prototype are inverted to give, from [link], the desired inverse- Chebyshev filter transfer function of
Equation:

$$F(s) = \frac{s^2 + 4/3}{(s + 0.85345)\left(s^2 + 0.57425s + 0.490095\right)}$$

Elliptic-Function Filter Properties

In this section, a design procedure is developed that uses a Chebyshev error criterion in both the passband and the stopband. This is the fourth possible combination of Chebyshev and Taylor's series approximations in the passband and stopband. The resulting filter is called an elliptic-function filter, because elliptic functions are normally used to calculate the pole and zero locations. It is also sometimes called a Cauer filter or a rational Chebyshev filter, and it has equal ripple approximation error in both pass and stopbands [link], [link], [link], [link].

The error criteria of the elliptic-function filter are particularly well suited to the way specifications for filters are often given. For that reason, use of the elliptic-function filter design usually gives the lowest order filter of the four classical filter design methods for a given set of specifications. Unfortunately, the design of this filter is the most complicated of the four. However, because of the efficiency of this class of filters, it is worthwhile gaining some understanding of the mathematics behind the design procedure.

This section sketches an outline of the theory of elliptic- function filter design. The details and properties of the elliptic functions themselves should simply be accepted, and attention put on understanding the overall picture. A more complete development is available in [link], [link]. Straightforward design of elliptic-function filters can be accomplished by skipping this section and going directly to Program 8 in the appendix or by using Matlab. However, it is important to understand the basics of the underlying theory to use the packaged design programs intelligently.

Because both the passband and stopband approximations are over the entire bands, a transition band between the two must be defined. Using a normalized passband edge, the bands are defined by
Equation:

$$0 < \omega < 1 \quad \text{passband}$$

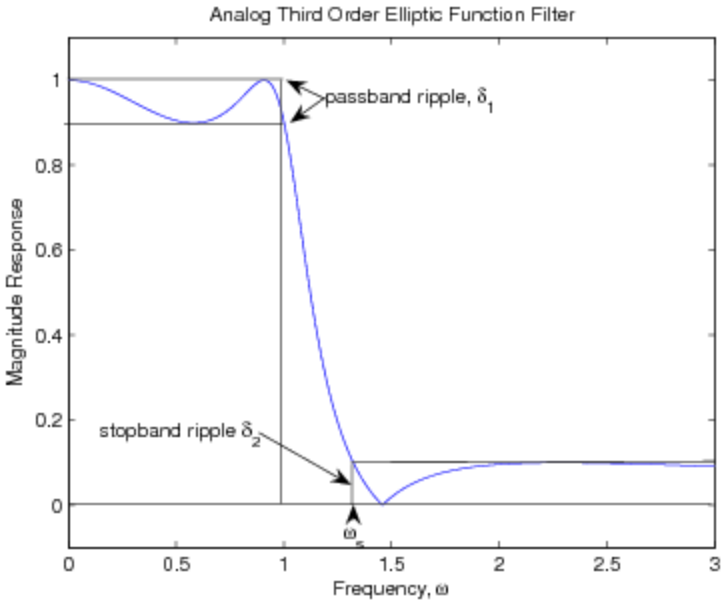
Equation:

$$1 < \omega < \omega_s \quad \text{transition band}$$

Equation:

$$\omega_s < \omega < \infty \quad \text{stopband}$$

This is illustrated in [link].



Third Order Analog Elliptic Function Lowpass Filter showing the Ripples and Band Edges

The characteristics of the elliptic function filter are best described in terms of the four parameters that specify the frequency response:

1. The maximum variation or ripple in the passband δ_1,
2. The width of the transition band $(\omega_s - 1)$,
3. The maximum response or ripple in the stopband δ_2, and

4. The order of the filter N.

The result of the design is that for any three of the parameters given, the fourth is minimum. This is a very flexible and powerful description of a filter frequency response.

The form of the frequency-response function is a generalization of that for the Chebyshev filter
Equation:

$$FF\left(j\omega\right) = \left|F\left(j\omega\right)\right|^2 = \frac{1}{1 + \epsilon^2 G^2\left(\omega\right)}$$

where
Equation:

$$FF(s) = F(s)F(-s)$$

with $F(s)$ being the prototype analog filter transfer function similar to that for the Chebyshev filter. $G(\omega)$ is a rational function that approximates zero in the passband and infinity in the stopband. The definition of this function is a generalization of the definition of the Chebyshev polynomial.

Elliptic Functions

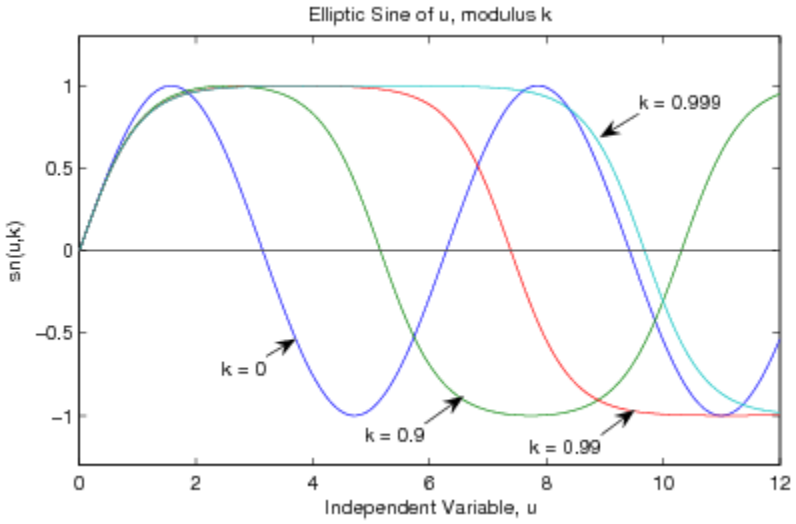
In order to develop analytical expressions for equal-ripple rational functions, an interesting class of transcendental functions, called the Jacobian elliptic functions, is outlined. These functions can be viewed as a generalization of the normal trigonometric and hyperbolic functions. The elliptic integral of the first kind [link] is defined as
Equation:

$$u\left(\varphi, k\right) = \int_0^{\varphi} \frac{dy}{\sqrt{1 - k^2\,\sin^2\left(y\right)}}$$

The trigonometric sine of the inverse of this function is defined as the Jacobian elliptic sine of u with modulus k, and is denoted
Equation:

$$sn(u,k) = \sin\left(\varphi(u,k)\right)$$

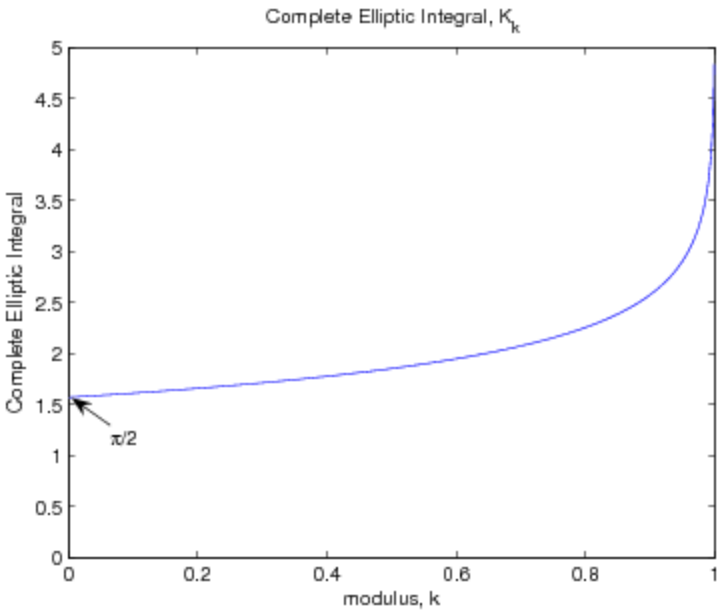
A special evaluation of [link] is known as the complete elliptic integral $K = u(\pi/2, k)$. It can be shown [link] that $sn(u)$ and most of the other elliptic functions are periodic with periods $4K$ if u is real. Because of this, K is also called the “quarter period". A plot of $sn(u,k)$ for several values of the modulus k is shown in [link].



Jacobian Elliptic Sine Function of u with Modulus k

For k=0, $sn(u,0) = \sin\left(u\right)$. As k approaches 1, the $sn(u,k)$ looks like a "fat" sine function. For $k = 1$, $sn(u,1) = \tanh\left(u\right)$ and is not periodic (period becomes infinite).

The quarter period or complete elliptic integral K is a function of the modulus k and is illustrated in [link].



Complete Elliptic Integral as a function of the Modulus

For a modulus of zero, the quarter period is $K = \pi/2$ and it does not increase much until k nears unity. It then increases rapidly and goes to infinity as k goes to unity.

Another parameter that is used is the complementary modulus k' defined by
Equation:

$$k^2 + k'^2 = 1$$

where both k and k' are assumed real and between 0 and 1. The complete elliptic integral of the complementary modulus is denoted K'.

In addition to the elliptic sine, other elliptic functions that are rather obvious generalizations are
Equation:

$$cn(u,k) = cos(\varphi(u,k))$$

Equation:

$$sc(u,k) = tan(\varphi(u,k))$$

Equation:

$$cs(u,k) = ctn(\varphi(u,k))$$

Equation:

$$nc(u,k) = sec(\varphi(u,k))$$

Equation:

$$ns(u,k) = csc(\varphi(u,k))$$

There are six other elliptic functions that have no trigonometric counterparts [link]. One that is needed is
Equation:

$$dn\left(u,k\right) = \sqrt{1-k^2sn^2\left(u,k\right)}$$

Many interesting properties of the elliptic functions exist [link]. They obey a large set of identities such as
Equation:

$$sn^2\left(u,k\right) + cn^2\left(u,k\right) = 1$$

They have derivatives that are elliptic functions. For example,
Equation:

$$\frac{d\ sn}{du} = cn\ dn$$

The elliptic functions are the solutions of a set of nonlinear differential equations of the form
Equation:

$$x'' + ax \pm bx^3 = 0$$

Some of the most important properties for the elliptic functions are as functions of a complex variable. For a purely imaginary argument
Equation:

$$sn\left(jv, k\right) = jsc\left(v, k'\right)$$

Equation:

$$cn\left(jv, k\right) = nc\left(v, k'\right)$$

This indicates that the elliptic functions, in contrast to the circular and hyperbolic trigonometric functions, are periodic in both the real and the imaginary part of the argument with periods related to K and K', respectively. They are the only class of functions that are “doubly periodic".

One particular value that the sn function takes on that is important in creating a rational function is
Equation:

$$sn\left(K + jK', k\right) = 1/k$$

The Chebyshev Rational Function

The rational function $G(\omega)$ needed in [link] is sometimes called a Chebyshev rational function because of its equal-ripple properties. It can be defined in terms of two elliptic functions with moduli k and k_1 by
Equation:

$$G\left(\omega\right) = sn\ \left(n\ sn^{-1}\left(\omega, k\right), k_1\right)$$

In terms of the intermediate complex variable φ, $G(\omega)$ and ω become
Equation:

$$G\left(\omega\right) = sn\left(n\varphi, k_1\right)$$

Equation:

$$\omega = sn(\varphi, k)$$

It can be shown [link] that $G(\omega)$ is a real-valued rational function if the parameters k, k_1, and n take on special values. Note the similarity of the definition of $G(\omega)$ to the definition of the Chebyshev polynomial $C_N\left(\omega\right)$. In this case, however, n is not necessarily an integer and is not the order of the filter. Requiring that $G(\omega)$ be a rational function requires an alignment of the imaginary periods [link] of the two elliptic functions in [link],[link]. It also requires alignment of an integer multiple of the real periods. The integer multiplier is denoted by N and is the order of the resulting filter [link]. These two requirements are stated by the following very important relations:
Equation:

$$nK' = K_1' \qquad \text{alignment of imaginary periods}$$

Equation:

$$nK = NK_1 \qquad \text{alignment of a multiple of the real periods}$$

which, on removing the parameter n, become
Equation:

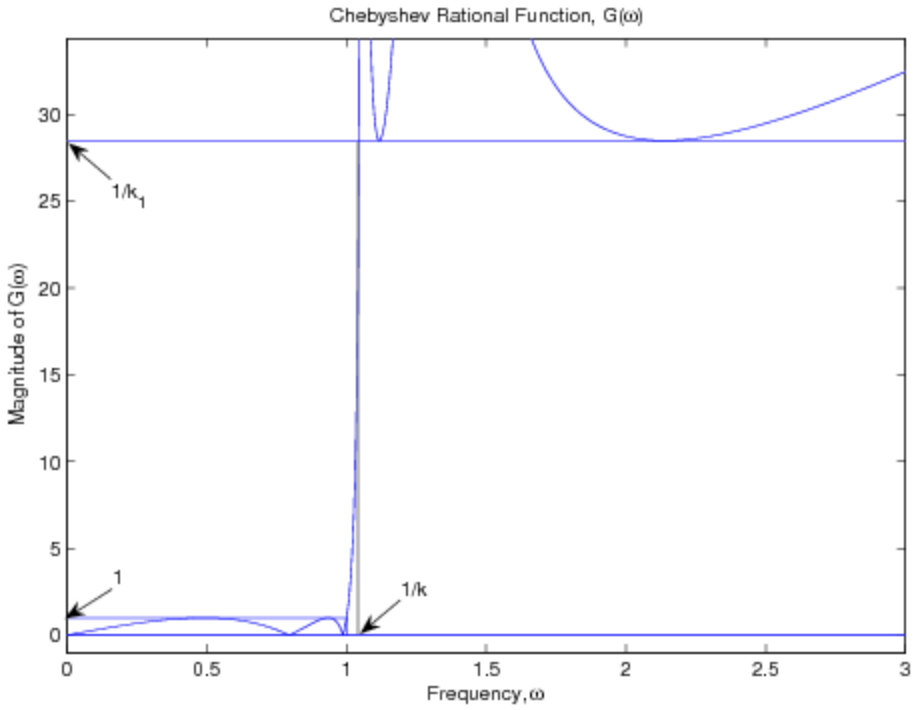
$$\frac{K_1}{K} N = \frac{K_1'}{K'}$$

or

Equation:

$$N = \frac{K K_1'}{K' K_1}$$

These relationships are central to the design of elliptic- function filters. N is an odd integer which is the order of the filter. For $N = 5$, the resulting rational function is shown in [link].



Fifth Order Elliptic Rational Function

This function is the basis of the approximation necessary for the optimal filter frequency response. It approximates zero over the frequency range $-1 < \omega < 1$ by an equal-ripple oscillation between ± 1. It also approximates infinity over the range $1/k < |\omega| < \infty$ by a reciprocal oscillation that keeps $|F(\omega)| > 1/k_1$. The zero approximation is normalized in both the frequency range and the $F(\omega)$ values to unity. The infinity approximation has its frequency range set by the choice of the modulus k, and the minimum value of $|F(\omega)|$ is set by the choice of the second modulus k_1.

If k and k_1 are determined from the filter specifications, they in turn determine the complementary moduli k' and k_1', which altogether determine the four values of the complete elliptic integral K needed to determine the order N in [link]. In general, this sequence of events will not result in an integer. In practice, however, the next larger integer is used, and either k or k_1 (or perhaps both) is altered to satisfy [link].

In addition to the two-band equal-ripple characteristics, $G(\omega)$ has another interesting and valuable property. The pole and zero locations have a reciprocal relationship that can be expressed by
Equation:

$$G(\omega)G(\omega_s/\omega) = 1/k_1$$

where
Equation:

$$\omega_s = 1/k$$

This states that if the zeros of $G(\omega)$ are located at ω_{zi}, the poles are located at
Equation:

$$\omega_{pi} = 1/(k\omega_{zi})$$

If the zeros are known, the poles are known, and vice versa. A similar relation exists between the points of zero derivatives in the 0 to 1 region and those in the $1/k$ to infinity region.

The zeros of $G(\omega)$ are found from [link] by requiring
Equation:

$$G\left(\omega\right) = sn\left[n\varphi, k_1\right] = 0$$

which implies
Equation:

$$n\varphi = 2K_1 i \text{for} i = 0,1,...$$

From [link], this gives
Equation:

$$\omega_{\text{zi}} = \text{sn}[2K_1 i/\text{n},\text{k}\ ],\ i = 0,1,...$$

This can be reformulated using [link] so that n and K_1 are not needed. For N odd, the zero locations are
Equation:

$$\omega_{\text{zi}} = \text{sn}[2K_1 i/\text{N},\text{k}\ ],\ i = 0,1,...$$

The pole locations are found from these zero locations using [link]. The locations of the zero-derivative points are given by
Equation:

$$\omega_{di} = sn\left[K\left(2i+1\right)/N, k\right]$$

in the 0 to 1 region, and the corresponding points in the 1/k to infinity region are found from [link].

The above relations assume N to be an odd integer. A modification for N even is necessary. For proper alignment of the real periods, the original definition of $G(\omega)$ is changed to

Equation:

$$G\left(\omega\right) = sn\left[\varphi + K_1, k_1\right]$$

which gives for the zero locations with N even

Equation:

$$\omega_{zi} = sn\left[(2i+1)K_1/n, k\right]$$

The even and odd N cases can be combined to give

Equation:

$$\omega_{zi} = \pm sn\left(iK/N, k\right)$$

for

Equation:

$$i = 0, 2, 4, \ldots, N-1 \quad \text{for N odd}$$

Equation:

$$i = 1, 3, 5, \ldots, N-1 \quad \text{for N even}$$

with the poles determined from [link].

Note that it is possible to determine $G(\omega)$ from k and N without explicitly using k_1 or n. Values for k_1 and n are implied by the requirements of [link] or [link].

Zero Locations

The locations of the zeros of the filter transfer function $F(\omega)$ are easily found since they are the same as the poles of $G(\omega)$, given in [link].
Equation:

$$\omega_{zi} = \frac{\pm 1}{k \ sn(iK/N, k)}$$

for
Equation:

$$i = 0, 2, 4, \ldots, N-1 \quad \text{N odd}$$

Equation:

$$i = 1, 3, 5, \ldots, N-1 \quad \text{N even}$$

These zeros are purely imaginary and lie on the ω axis.

Pole Locations

The pole locations are somewhat more complicated to find. An approach similar to that used for the Chebyshev filter is used here. $FF(s)$ becomes infinite when
Equation:

$$1 + \epsilon^2 G^2 = 0$$

or
Equation:

$$G = \pm j(1/\epsilon)$$

Using [link] and the periodicity of sn (u,k) , this implies
Equation:

$$G = sn(n\varphi + 2K_1 i, k_1) = \pm j1/\epsilon$$

or
Equation:

$$\varphi = \left(-2K_1 i + sn^{-1}\left(j1/e, k_1\right)\right)/n$$

Define ν_0 to be the second term in [link] by
Equation:

$$j\nu_0 = \left(sn^{-1}\left(j1/e, k_1\right)\right)/n$$

which is similar to the equation for the Chebyshev case. Using properties of sn of an imaginary variable and [link], ν_0 becomes
Equation:

$$\nu_0 = (K/NK_1)sc^{-1}\left(1/\epsilon, k'\right)$$

The poles are now found from [link],[link], [link], and [link] to be
Equation:

$$s_{pi} = j\ sn\left(Ki/N + j\nu_0, k\right)$$

This equation can be more clearly written by using the summation formula [link] for the elliptic sine function to give
Equation:

$$s_{pi} = \frac{cn\ dn\ sn'\ cn' + jsn\ dn'}{1 - dn^2 sn'^2}$$

where
Equation:

$$sn = sn(Ki/N, k), \quad cn = cn(Ki/N, k), \quad dn = dn(Ki/N, k)$$

Equation:

$$sn' = sn\left(\nu_0, k'\right), \quad cn' = cn\left(\nu_0, k'\right), \quad dn' = dn\left(\nu_0, k'\right)$$

for
Equation:

$$i = 0, 2, 4, \ldots . \quad \text{N odd}$$

Equation:

$$i = 1, 3, 5, \ldots . \quad \text{N even}$$

The theory of Jacobian elliptic functions can be found in [link] and its application to filter design in [link], [link], [link]. The best techniques for calculating the elliptic functions seem to use the arithmetic-geometric mean; efficient algorithms are presented in [link]. A design program is given in [link] and a versitile FORTRAN program that is easily related to the theory in this chapter is given as Program 8 in the appendix of this book. Matlab has a powerful elliptic function filter design command as well as accurate algorithms for evaluating the Jacobian elliptic functions and integrals.

An alternative to the use of elliptic functions for finding the transfer function $F(s)$ pole locations is to obtain the zeros from [link], then find $G(\omega)$ using the reciprocal relation of the poles and zeros [link]. $F(s)$ is constructed from $G(\omega)$ and ϵ from [link], and the poles are found using a root-finding algorithm. Another possibility is to find the zeros from [link] and the poles from the methods for finding a Chebyshev passband from arbitrary zeros. These approaches avoid calculating ν_0 by [link] or determining k from K/K', as is described in [link]. The efficient algorithms for evaluating the elliptic functions and the common use of powerful computers make these alternatives less attractive now.

Summary

In this section the basic properties of the Jacobian elliptic functions have been outlined and the necessary conditions given for an equal-ripple rational function to be defined in terms of them. This rational function was then used to construct a filter transfer function with equal-ripple properties. Formulas were derived to calculate the pole and zero locations for the filter transfer functions and to relate design specifications to the functions. These formulas require the evaluation of elliptic functions and are implemented in Program 8 in the appendix.

Elliptic-Function Filter Design Procedures

The equal-ripple rational function $G(\omega)$ is used to describe an optimal frequency-response function $F(j\omega)$ and to design the corresponding filter. The squared-magnitude frequency-response function is

Equation:

$$|F(j\omega)|^2 = \frac{1}{1+\epsilon^2 G(\omega)^2}$$

with $G(\omega)$ defined by Jacobian Elliptic functions, and ϵ being a parameter that controls the passband ripple. The plot of this function for $N = 3$ illustrates the relation to the various specification parameters.

From [link], it is seen that the passband ripple is measured by δ_1, the stopband ripple by δ_2, and the normalized transition band by ω_s. The previous section showed that

Equation:

$$\omega_s = 1/k$$

which means that the width of the transition band determines k. It should be remembered that this development has assumed a passband edge

normalized to unity. For the unnormalized case, the passband edge is ω_p and the stopband edge becomes
Equation:

$$\omega_s = \frac{\omega_p}{k}$$

The stopband performance is described in terms of the ripple δ_2 normalized to a maximum passband response of unity, or in terms of the attenuation b in the stopband expressed in positive dB assuming a maximum passband response of zero dB. The stopband ripple and attenuation are determined from [link] and [link] to be
Equation:

$$\delta_2^2 = 10^{-b/10} = \frac{1}{1 + \epsilon^2 / k_1^2}$$

This can be rearranged to give k_1 in terms of the stopband ripple or attenuation.
Equation:

$$k_1^2 = \frac{\epsilon^2}{1/\delta_2^2 - 1} = \frac{\epsilon^2}{10^{b/10} - 1}$$

The order N of the filter depends on k and k_1, as shown in [link]. Equations [link], [link], and [link] determine the relation of the frequency-response specifications and the elliptic-function parameters. The location of the transfer function poles and zeros must then be determined.

Because of the required relationships of [link] and the fact that the order N must be an integer, the passband ripple, stopband ripple, and transition band cannot be independently set. Several straightforward procedures can be used that will always meet two of the specifications and exceed the third.

The first design step is generally the determination of the order N from the desired passband ripple δ_1, the stopband ripple δ_2, and the transition band controlled by ω_s. The following formulas determine the moduli k and k_1 from the passband ripple δ_1 or its dB equavilent a, and the stopband ripple δ_2 or its dB attenuation equivalent b:

Equation:

$$\epsilon = \sqrt{\frac{2\delta_1 - \delta_1^2}{1 - 2\delta_1 - \delta_1^2}} = \sqrt{10^{a/10} - 1}$$

Equation:

$$k_1 = \frac{\epsilon}{\sqrt{1/\delta_2^2 - 1}} = \frac{\epsilon}{\sqrt{10^{b/10} - 1}}$$

Equation:

$$k_1' = \sqrt{1 - k_1^2}$$

Equation:

$$k = \omega_p/\omega_s \qquad k' = \sqrt{1 - k^2}$$

The order N is the smallest integer satisfying

Equation:

$$N \geq \frac{KK_1'}{K'K_1}$$

This integer order N will not in general exactly satisfy [link], i.e., will not satisfy [link] with equality. Either k or k_1 must to recalculated to satisfy [link] and [link]. The various possibilities for this are developed below.

Methods for Meeting Specifications

Fixed Order, Passband Ripple, and Transition Band

Given N from [link] and the specifications δ_1, ω_p, and ω_s, the parameters ϵ and k are found from [link] and (refcc50). From k, the complete elliptic integrals K and K' are calculated [link]. From [link], the ratio K/K' determines the ratio K_1'/K_1. Using numerical methods from [link], k_1 is calculated. This gives the desired δ_1, ω_p, and ω_s and minimizes the stopband ripple δ_2 (or maximizes the stopband attenuation b).

Using these parameters, the zeros are calculated from (refcc31) and the poles from (refcc39). Note the zero locations do not depend on ϵ or k_1, but only on N and ω_s. This makes the tradeoff between stop and passband occur in (refcc48) and only affects the calculation of nu_0 in (refcc38)

This approach which minimizes the stopband ripple is used in the IIR filter design program in the appendix of this book.

Fixed Order, Stopband Rejection, and Transition Band

Given N from [link] and the specifications δ_2, ω_p, and ω_s, the parameter k is found from (refcc50). From k, the complete elliptic integrals K and K' are calculated [link]. From [link], the ratio K/K' determines the ratio K_1'/K_1 . Using numerical methods from [link], k_1 is calculated. From k_1 and δ_2, ϵ and δ_1 are found from

Equation:

$$\epsilon = k_1\sqrt{1/\delta_2^2 - 1}$$

and

Equation:

$$\delta_1 = 1 - \frac{1}{\sqrt{1+\epsilon^2}}$$

This set of parameters gives the desired ω_p, ω_s, and stopband ripple and minimizes the passband ripple. The zero and pole locations are found as above.

Fixed Order, Stopband, and Passband Ripple

Given N from [link] and the specifications δ_1, δ_2, and either ω_p or ω_s, the parameters ϵ and k_1 are found from [link] and (refcc48). From k_1, the complete elliptic integrals K_1 and K_1' are calculated [link]. From [link], the ratio K_1/K_1' determines the ratio K'/K. Using numerical methods from [link], k is calculated. This gives the desired passband and stopband ripple and minimizes the transition-band width. The pole and zero locations are found as above.

An Approximation

In many filter design programs, after the order N is found from [link], the design proceeds using the original e, k, and k_1, even though they do not satisfy [link]. The resulting design has the desired transition band, but both pass and stopband ripple are smaller than specified. This avoids the calculation of the modulus k or k_1 from a ratio of complete elliptic integrals as was necessary in all three cases above, but produces results that are difficult to exactly predict.

Example:
Design of a Third-Order Elliptic-Function Filter
A lowpass elliptic-function filter is desired with a maximum passband ripple of $\delta_1 = 0.1$ or $a = 0.91515$ dB, a maximum stopband ripple of $\delta_2 = 0.1$ or $b = 20$ dB rejection, and a normalized stopband edge of

$\omega_s = 1.3$ radians per second. The first step is to determine the order of the filter.
From ω_s, the modulus k is calculated and then the complimentary modulus using the relations in (refcc50). Special numerical algorithms illustrated in Program 8 are then used to find the complete elliptic integrals K and K' [link].
Equation:

$$k = 1/1.3 = 0.769231, \qquad k' = \sqrt{1 - k^2} = 0.638971$$

Equation:

$$K = 1.940714, \qquad K' = 1.783308$$

From δ_1, ϵ is calculated using [link], and from ϵ and δ_2, k_1 is calculated from (refcc48). k_1', K_1, and K_1' are then calculated.
Equation:

$$\epsilon = 0.4843221 \qquad \text{as for the Chebyshev example.}$$

Equation:

$$k_1 = 0.0486762, \qquad k_1' = 0.9988146$$

Equation:

$$K_1 = 1.571727, \qquad K_1' = 4.4108715$$

The order is obtained from [link] by calculating
Equation:

$$\frac{K \ K'}{K' \ K_1} = 3.0541$$

This is close enough to 3 to set $N = 3$. Rather than recalculate k and k_1, the already calculated values are used as discussed in the design method D in this section. The zeros are found from (refcc31) using only N and k from above.

Equation:

$$\omega_z = \frac{\pm 1}{k\ sn(2K/N, k)} = \pm 1.430207$$

To find the pole locations requires the calculation of ν_0 from (refcc38) which is somewhat complicated. It is carried out using the algorithms in Program 8 in the appendix.
Equation:

$$\nu_0 = \frac{K}{N\ K_1} sc^{-1}\left(1/\epsilon, k_1'\right) = 0.6059485$$

From this value of ν_0, and k and N above, the elliptic functions in (refcc40) are calculated to give
Equation:

$$sn' = .557986, \quad cn' = 0.829850, \quad dn' = 0.934281$$

which, for the single real pole corresponding to i = 0 in (refcc39), gives
Equation:

$$s_p = 0.672393$$

For the complex conjugate pair of poles corresponding to $i = 2$, the other elliptic functions in (refcc40) are
Equation:

$$sn = 0.908959, \quad cn = 0.416886, \quad dn = 0.714927$$

which gives from (refcc39) for the poles
Equation:

$$s_p = 0.164126 \pm j1.009942$$

The complete transfer function is
Equation:

$$F(s) = \frac{s^2 + 2.045492}{(s + 0.672393)\left(s^2 + 0.328252s + 1.046920\right)}$$

This design should be compared to the Chebyshev and inverse- Chebyshev designs.

Optimality of the Four Classical Filter Designs

It is important in designing filters to choose the particular type that is appropriate. Since in all cases, the filters are optimal, it is necessary to understand in what sense they are optimal.

The classical Butterworth filter is optimal in the sense that it is the best Taylor's series approximation to the ideal lowpass filter magnitude at both $\omega = 0$ and $\omega = \infty$.

The Chebyshev filter gives the smallest maximum magnitude error over the entire passband of any filter that is also a Taylor's series approximation at $\omega = \infty$ to the ideal magnitude characteristic.

The Inverse-Chebyshev filter is a Taylor's series approximation to the ideal magnitude response at $\omega = 0$ and minimizes the maximum error in the approximation to zero over the stopband. This can also be stated as maximizing the minimum rejection of the filter over the stopband.

The elliptic-function filter (Cauer filter) considers the four parameters of the filter: the passband ripple, the transition-band width, the stopband ripple, and the order of the filter. For given values of any three of the four, the fourth is minimized.

It should be remembered that all four of these filter designs are magnitude approximations and do not address the phase frequency response or the time-domain characteristics. For most designs, the Butterworth filter has the smoothest phase curve, followed by the inverse-Chebyshev, then the Chebyshev, and finally the elliptic-function filter having the least smooth phase response.

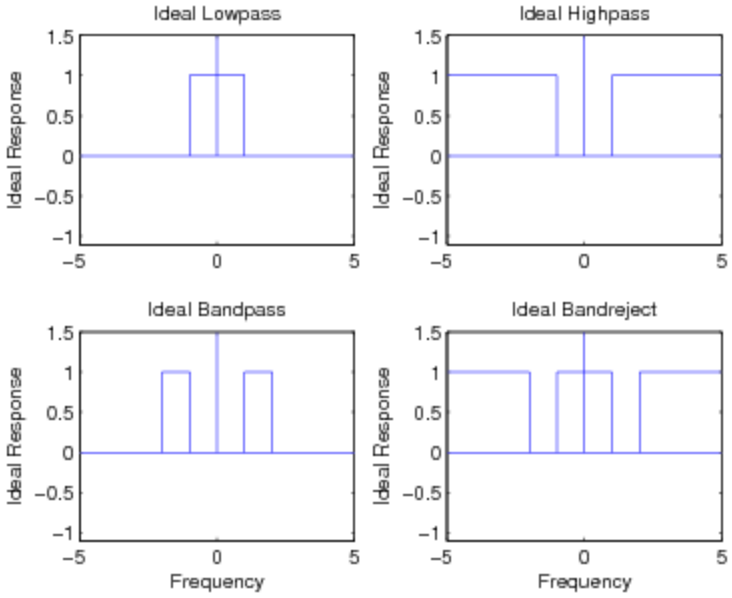
Recall that in addition to the four filters described in this section, the more general Taylor's series method allows an arbitrary zero locations to be specified but retains the optimal character at $\omega = 0$. A design similar to this can be obtained by replacing ω by $1/\omega$, which allows setting $|F(w)|^2$ equal unity at arbitrary frequencies in the passband and having a Taylor's series approximation to zero at $\omega = \infty$[link].

These basic normalized lowpass filters can have the passband edge moved from unity to any desired value by a simple change of frequency variable, ω replaced with $k\omega$. They can be converted to highpass filters or bandpass or band reject filters by various changes such as ω with k/ω or ω with $a\omega + b/\omega$. In all of these cases the optimality is maintained, because the basic lowpass approximation is to a piecewise constant ideal. An approximation to a nonpiecewise constant ideal, such as a differentiator, may not be optimal after a frequency change of variables .

In some cases, especially where time-domain characteristics are important, ripples in the frequency response cause irregularities, such as echoes in the time response. For that reason, the Butterworth and Chebyshev II filters are more desirable than their frequency response alone might indicate. A fifth approximation has been developed [link] that is similar to the Butterworth. It does not require a Taylor's series approximation at $\omega = 0$, but only requires that the response monotonically decrease in the passband, thus giving a narrower transition region than the Butterworth, but without the ripples of the Cheybshev.

Frequency Transformations

In addition to the lowpass frequency response, other basic ideal responses are often needed in practice. The ideal highpass filter rejects signals with frequencies below a certain value and passes those with frequencies above that value. The ideal bandpass filter passes only a band of frequencies, and the ideal band reject filter completely rejects a band of frequencies. These ideal frequency responses are illustrated in [link].



The Basic Four Ideal Frequency Responses

This section presents a method for designing the three new filters by using a frequency transformation on the basic lowpass design. When used on the four classical IIR approximations (e.g. Butterworth, Chebyshev, inverse-Chebyshev, and Elliptic Function), the optimality is preserved. This procedure is used in the FREQXFM() subroutine of Program 8 in the appendix.

Change the Bandedge

The classical filters have all been developed for a bandedge of $\omega_0 = 1$. That is where the Butterworth filter has a magnitude squared of one half: $|F| = 0.5$ or the Chebyshev filter has its passband edge or the Inverse Chebyshev has its stopband edge or the Elliptic filter has its passband edge. To scale the bandedge, simply replace s by Ks or: $s \to Ks$ where K is reciprocal of the new desired bandedge. What happened to the prototype filter at $\omega = 1$ will now happen at $\omega = 1/K$. It is simply a linear scaling of the ω axis. This change can be done before the conversions below or after.

The Highpass Filter

The frequency response illustrated in [link]b can be obtained from that in [link]a by replacing the complex frequency variable s in the transfer function by $1/s$. This change of variable maps zero frequency to infinity, maps unity into unity, and maps infinity to zero. It turns the complex s plane inside out and leaves the unit circle alone.

In the design procedure, the desired bandedge ω_0 for the highpass filter is mapped by $1/\omega_0$ to give the bandedge for the prototype lowpass filter. This lowpass filter is next designed by one of the optimal procedures already covered and then converted to a highpass transfer function by replacing s by $1/s$. If an elliptic-function filter approximation is used, both the passband edge ω_p and the stopbandedge ω_s are transformed. Because most optimal lowpass design procedures give the designed transfer function in factored form from explicit formulas for the poles and zeros, the transformation can be performed on each pole and zero to give the highpass transfer function in factored form.

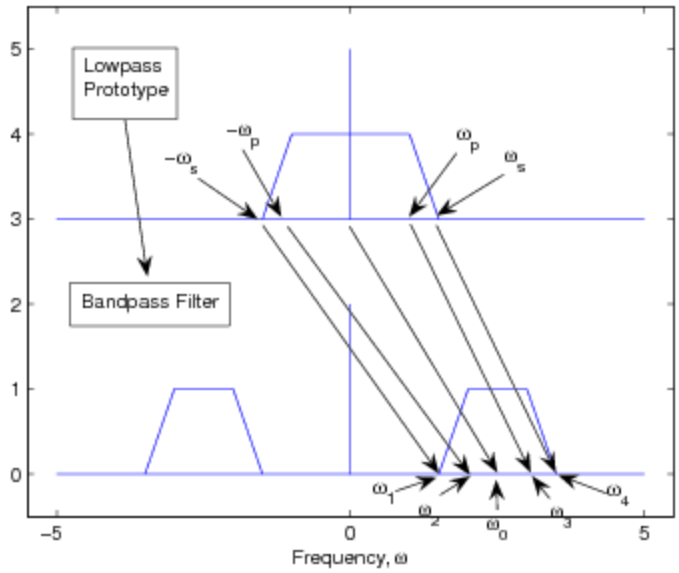
The Bandpass Filter

In order to convert the lowpass filter of [link]a into that of [link]c, a more complicated frequency transformation is required. In order to reduce confusion, the complex frequency variable for the prototype analog filter transfer function will be denoted by p and that for the transformed analog filter by s. The transformation is given by

Equation:

$$p = \frac{s^2 + \omega_0^2}{s}$$

This change of variables doubles the order of the filter, maps the origin of the s-plane to both plus and minus $j\omega_0$, and maps minus and plus infinity to zero and infinity. The entire ω axis of the prototype response is mapped between zero and plus infinity on the transformed responses. It is also mapped onto the left-half axis between minus infinity and zero. This is illustrated in [link].



Lowpass to Bandpass Transformation

In order that the transformation give $-\omega_p = \left(\omega_2^2 - \omega_0^2\right)/\omega_2$ and $\omega_p = \left(\omega_3^2 - \omega_0^2\right)/\omega_3$, the “center" frequency ω_0 must be
Equation:

$$\omega_0 = \sqrt{\omega_2 \omega_3}$$

However, because $-\omega_s = \left(\omega_1^2 - \omega_0^2\right)/\omega_1$ and $\omega_s = \left(\omega_4^2 - \omega_0^2\right)/\omega_4$, the center frequency must also be
Equation:

$$\omega_0 = \sqrt{\omega_1 \omega_4}$$

This means that only three of the four bandedge frequencies ω_1, ω_2, ω_3, and ω_4 can be independently specified. Normally, ω_0 is determined by ω_2 and ω_3 which then specifies the prototype passband edge by
Equation:

$$\omega_p = \frac{\omega_3^2 - \omega_0^2}{\omega_3}$$

and, using the same ω_0, the stopband edge is set by either ω_1 or ω_4, whichever gives the smaller ω_s.
Equation:

$$\omega_s = \frac{\omega_4^2 - \omega_0^2}{\omega_4} \quad \text{or} \quad \frac{\omega_0^2 - \omega_1^2}{\omega_1}$$

The finally designed bandpass filter will meet both passband edges and one transition band width, but the other will be narrower than originally specified. This is not a problem with the Butterworth or either of the Chebyshev approximation because they only have passband edges or stopband edges, but not both. The elliptic-function has both.

After the bandedges for the prototype lowpass filter ω_p and/or ω_s are calculated, the filter is designed by one of the optimal approximation methods discussed in this section or any other means. Because most of these methods give the pole and zero locations directly, they can be individually transformed to give the bandpass filter transfer function in factored form. This is accomplished by solving $s^2 - ps + \omega_0^2$ from the original transformation to give for the root locations

Equation:

$$s = \frac{p \pm \sqrt{p^2 - 4\omega_0^2}}{2}$$

This gives two transformed roots for each prototype root which doubles the order as expected.

The roots that result from transforming the real pole of an odd- order prototype cause some complication in programming this procedure. Program 8 should be studied to understand how this is carried out.

The Band-Reject Filter

To design a filter that will reject a band of frequencies, a frequency transformation of the form
Equation:

$$p = \frac{s}{s^2 + \omega_0^2}$$

is used on the prototype lowpass filter. This transforms the origin of the p-plane into both the origin and infinity of the s-plane. It maps infinity in the p-plane into jω_0 in the s-plane.

Similar to the bandpass case, the transformation must give $-\omega_p = \omega_4 / \left(\omega_0^2 - \omega_4^2\right)$ and $\omega_p = \omega_1 / \left(\omega_0^2 - \omega_1^2\right)$. A similar relation of ω_s to ω_2 and ω_3 requires that the center frequency ω_0 must be
Equation:

$$\omega_0 = \sqrt{\omega_1 \omega_4} = \sqrt{\omega_2 \omega_3}$$

As before, only three of the four bandedge frequencies can be independently specified. Normally, ω_0 is determined by ω_1 and ω_4 which

then specifies the prototype passband edge by
Equation:

$$\omega_p = \frac{\omega_1}{\omega_0^2 - \omega_1^2}$$

and, using the same ω_0, the stopband edge is set by either ω_2 or ω_3, whichever gives the smaller ω_s.
Equation:

$$\omega_s = \frac{\omega_2}{\omega_0^2 - \omega_2^2} \quad \text{or} \quad \frac{\omega_3}{\omega_4^2 - \omega_0^2}$$

The finally designed bandpass filter will meet both passband edges and one transition-band width, but the other will be narrower than originally specified. This does not occur with the Butterworth or either Chebyshev approximation, only with the elliptic-function.

After the bandedges for the prototype lowpass filter ω_p and/or ω_s are calculated, the filter is designed. The poles and zeros of this filter are individually transformed to give the bandreject filter transfer function in factored form. This is carried out by solving $s^2 - (1/p)s + \omega_0^2$ to give for the root locations
Equation:

$$s = \frac{1/p \pm \sqrt{(1/p)^2 - 4\omega_0^2}}{2}$$

A more complicated set of transformations could be developed by using a general map of $s = f(s)$ with a higher order. Several pass or stopbands could be specified, but the calculations become fairly complicated.

Although this method of transformation is a powerful and simple way for designing bandpass and bandreject filters, it does impose certain

restrictions. A Chebyshev bandpass filter will be equal-ripple in the passband and maximally flat at both zero and infinity, but the transformation forces the degree of flatness at zero and infinity to be equal. The elliptic-function bandpass filter will bave the same number of ripples in both stopbands even if they are of very different widths. These restrictions are usually considered mild when compared with the complexity of alternative design methods.

Conversion of Analog to Digital Transfer Functions

For mathematical convenience, the four classical IIR filter transfer functions were developed in terms of the Laplace transform rather than the z-transform. The prototype Laplace-transform transfer functions are descriptions of analog filters. In this section they are converted to z-transform transfer functions for implementation as IIR digital filters.

There have been several different methods of converting analog systems to digital described over the history of digital filters. Two have proven to be useful for most applications. The first is called the impulse-invariant method and results in a digital filter with an impulse response exactly equal to samples of the prototype analog filter. The second method uses a frequency mapping to convert the analog filter to a digital filter. It has the desirable property of preserving the optimality of the four classical approximations developed in the last section. This section will develop the theory and design formulas to implement both of these conversion approaches.

The Impulse-Invariant Method

Although the transfer functions in Continuous Frequency Definition of Error were designed with criteria in the frequency domain, the impulse-invariant method will convert them into digital transfer functions using a time-domain constraint [link], [link], [link]. The digital filter designed by the impulse-invariant method is required to have an impulse response that is exactly equal to equally spaced samples of the impulse response of the prototype analog filter. If the analog filter has a transfer function $F(s)$ with an impulse response $f(t)$, the impulse response of the digital filter $h(n)$ is required to match the samples of $f(t)$. For samples at T second intervals, the impulse response is

Equation:

$$h\left(n\right) = F\left(T\right)|_{t=Tn} = F\left(Tn\right)$$

The transfer function of the digital filter is the z-transform of the impulse response of the filter, which is given by

Equation:

$$H\left(z\right) = \sum_{n=0}^{\infty} h\left(n\right) z^{-n}$$

The transfer function of the prototype analog filter is always a rational function written as
Equation:

$$F(s) = \frac{B(s)}{A(s)}$$

where $B(s)$ is the numerator polynomial with roots that are the zeros of $F(s)$, and $A(s)$ is the denominator with roots that are the poles of $F(s)$. If $F(s)$ is expanded in terms of partial fractions, it can be written as
Equation:

$$F(s) = \sum_{i=1}^{N} \frac{K_i}{s + s_i}$$

The impulse response of this filter is the inverse-Laplace transform of [link], which is
Equation:

$$f(t) = \sum_{i=1}^{N} K \ e^{s_i t}$$

Sampling this impulse response every T seconds gives
Equation:

$$f(nT) = \sum_{i=1}^{N} K_i \ e^{-s_i nT} = \sum_{i=1}^{N} K_i \left(e^{-s_i T}\right)^n$$

The basic requirement of [link] gives
Equation:

$$H(z) = \sum_{n=0}^{\infty} \left[\sum_{i=1}^{N} K_i \left(e^{-sIT}\right)^n \right]$$

Equation:

$$H(z) = \sum_{i=1}^{N} \frac{K_i z}{z - e^{sIT}}$$

which is clearly a rational function of z and is the transfer function of the digital filter, which has samples of the prototype analog filter as its impulse response.

This method has its requirements set in the time domain, but the frequency response is important. In most cases, the prototype analog filter is one of the classical types, which is optimal in the frequency domain. If the frequency response of the analog filter is denoted by $F(j\omega)$ and the frequency response of the digital filter designed by the impulse- invariant method is $H(\omega)$, it can be shown in a development similar to that used for the sampling theorem
Equation:

$$H(\omega) = (1/T) \sum_{k=-\infty}^{\infty} F(j(\omega - 2\pi k/T))$$

The frequency response of the digital filter is a periodically repeated version of the frequency response of the analog filter. This results in an overlapping of the analog response, thus not preserving optimality in the same sense the analog prototype was optimal. It is a similar phenomenon to the aliasing that occurs when sampling a continuous-time signal to obtain a digital signal in A-to-D conversion. If $F(j\omega)$ is an analog lowpass filter that goes to zero as ω goes to infinity, the effects of the folding can be made small by high sampling rates (small T).

The impulse-invariant design method can be summarized in the following steps:

1. Design a prototype analog filter with transfer function $F(j\omega)$.
2. Make a partial fraction expansion of $F(j\omega)$ to obtain the N values for K_i and s_i .
3. Form the digital transfer function $H(z)$ from [link] to give the desired design.

The characteristics of the designed filter are the following:

- It has N poles, the same as the analog filter.
- It is stable if the analog filter was stable. This is seen from the change of variables in the denominator of (6.70) which maps the left-half s-plane inside the unit circle in the z-plane.

- The frequency response is a folded version of the analog filter, and the optimal properties of the analog filter are not preserved.
- The cascade of two impulse-invariant designed filters are not impulse-invariant with the cascade of the two analog prototypes. In other words, the filter must be designed in one step.

This method is sometimes used to design digital filters, but because the relation of the analog and digital system is specified in the time domain, it is more useful in designing a digital simulation of an analog system. Unfortunately, the properties of this class of filters depend on the input. If a filter is designed so that its impulse response is the sampled impulse response of the analog filter, its step response will not be the sampled step response of the analog filter.

A step-invariant filter can be designed by first multiplying the analog filter transfer function $F(s)$ by $1/s$, which is the Laplace transform of a step function. This product is then expanded in partial fraction just as $F(s)$ was in [link] and the same substitution made as in [link] giving a z-transform. After the z-transform of a step is removed, the digital filter has the step-invariant property. This idea can be extended to other input functions, but the impulse-invariant version is the most common. Another modification to the impulse-invariant method is known as the matched z transform covered in [link], but it is less useful.

There can be a problem with the classical impulse-invariant method when the number of finite zeros is too large. This is addressed in [link], [link].

An example of a Butterworth lowpass filter used to design a digital filter by the impulse-invariant method can be shown. Note that the frequency response does not go to zero at the highest frequency of $w = p$. It can be made as small as desired by increasing the sampling rate, but this is more expensive to implement. Because the frequency response of the prototype analog filter for an inverse-Chebyshev or elliptic-function filter does not necessarily go to zero as w goes to infinity, the effects of folding on the digital frequency response are poor. No amount of sampling rate increase will change this. The same problem exists for a highpass filter. This shows the care that must be exercised in using the impulse-invariant design method.

The Bilinear Transformation

A second method for converting an analog prototype filter into a desired digital filter is the bilinear transformation. This method is entirely a frequency-domain method, and as a result, some of the optimal properties of the analog filter are preserved. As was the case with the impulse-invariant method, the time interval is not normalized to one, but is explicitly denoted by the sampling interval T with units of seconds. The

bilinear transformation is a change of variables (a mapping) that is linear in both the numerator and denominator [link], [link], [link], [link]. The usual form is
Equation:

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

The z-transform transfer function of the digital filter $H(z)$ is obtained from the Laplace transform transfer function $F(s)$ of the prototype filter by substituting for s the bilinear form of [link].
Equation:

$$H\left(z\right) = F\left(\frac{2(z-1)}{T(z+1)}\right)$$

This operation can be reversed by solving [link] for z and substituting this into $H(z)$ to obtain $F(s)$. This reverse operation is also bilinear of the form
Equation:

$$z = \frac{2/T + s}{2/T - s}$$

To consider the frequency response, the Laplace variable s is evaluated on the imaginary axis and the z-transform variable z is evaluated on the unit circle. This is achieved by
Equation:

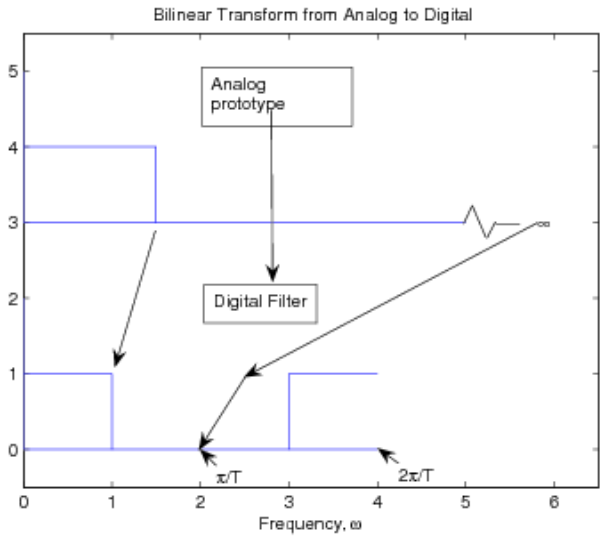
$$s = ju \quad \text{and} \quad z = e^{j\omega T}$$

which gives the relation of the analog frequency variable u to the digital frequency variable ω from [link] and [link] to be
Equation:

$$u = (2/T)\,\tan\left((\omega T)/2\right)$$

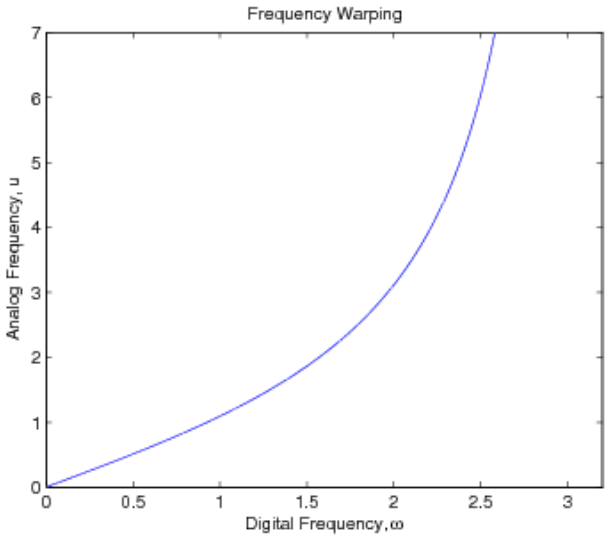
The bilinear transform maps the infinite imaginary axis in the-s plane onto the unit circle in the z-plane. It maps the infinite interval of $-\infty < u < \infty$ of the analog

frequency axis on to the finite interval of $-\pi/2 < \omega < \pi/2$ of the digital frequency axis. This is illustrated in [link].



The Frequency Map of the Bilinear Transform

There is no folding or aliasing of the prototype frequency response, but there is a compression of the frequency axis, which becomes extreme at high frequencies. This is shown in [link] from the relation of [link].



The Frequency Mapping of the Bilinear Transform

Near zero frequency, the relation of u and ω is essentially linear. The compression increases as the digital frequency w nears $\pi/2$. This nonlinear compression is called frequency warping. The conversion of $F(s)$ to $H(z)$ with the bilinear transformation does not change the values of the frequency response, but it changes the frequencies where the values occur.

In the design of a digital filter, the effects of the frequency warping must be taken into account. The prototype filter frequency scale must be prewarped so that after the bilinear transform, the critical frequencies are in the correct places. This prewarping or scaling of the prototype frequency scale is done by replacing s with Ks. Because the bilinear transform is also a change of variables, both can be performed in one step if that is desirable.

If the critical frequency for the prototype filter is u_o and the desired critical frequency for the digital filter is ω_o, the two frequency responses are related by
Equation:

$$F\left(ju_0\right) = H\left(\omega_0\right) = F^*$$

The prewarping scaling is given by
Equation:

$$u_0 = \frac{2}{T} \tan\left(\frac{\omega_0 T}{2}\right)$$

Combining the prewarping scale and the bilinear transformation give
Equation:

$$u_0 = \frac{2K}{T} \tan\left(\frac{\omega_0 T}{2}\right)$$

Solving for K and combining with [link] give
Equation:

$$s = \frac{u_0}{\tan\left(\omega_0 T/2\right)} \frac{z-1}{z+1}$$

All of the optimal filters developed in Continuous Frequency Definition of Error and most other prototype filters are designed with a normalized critical frequency of $u_0 = 1$. Recall that ω_0 is in radians per second. Most specifications are given in terms of frequency f in Hertz (cycles per second) which is related to ω or u by
Equation:

$$\omega = 2\pi f$$

Care must be taken with the elliptic-function filter where there are two critical frequencies that determine the transition region. Both frequencies must be prewarped.

The characteristics of the bilinear transform are the following:

- The order of the digital filter is the same as the prototype filter.
- The left-half s-plane is mapped into the unit circle on the z-plane. This means stability is preserved.
- Optimal approximations to piecewise constant prototype filters, such as the four cases in Continuous Frequency Definition of Error, transform into optimal digital filters.
- The cascade of sections designed by the bilinear transform is the same as obtained by transforming the total system.

The bilinear transform is probably the most used method of converting a prototype Laplace transform transfer function into a digital transfer function. It is the one used

in most popular filter design programs [link], because of characteristic 3 above that states optimality is preserved. The maximally flat prototype is transformed into a maximally flat digital filter. This property only holds for approximations to piecewise constant ideal frequency responses, because the frequency warping does not change the shape of a constant. If the prototype is an optimal approximation to a differentiator or to a linear-phase characteristic, the bilinear transform will destroy the optimality. Those approximations have to be made directly in the digital frequency domain.

Example:
The Bilinear Transformation
To illustrate the bilinear transformation, the third-order Butterworth lowpass filter designed in the Example is converted into a digital filter. The prototype filter transfer function is
Equation:

$$F(s) = \frac{1}{(s+1)(s^2+s+1)}$$

The prototype analog filter has a passband edge at $u_0 = 1$. A data rate of 1000 samples per second corresponding to $T = 0.001$ seconds is assumed. If the desired digital passband edge is $f_0 = 200$ Hz, then $\omega_0 = (2\pi)(200)$ radians per second, and the total prewarped bilinear transformation from [link] is
Equation:

$$s = 1.376382\frac{z-1}{z+1}$$

The digital transfer function in [link] becomes
Equation:

$$H(z) = \frac{0.09853116(z+1)^3}{(z-0.158384)(z^2-0.418856z+0.355447)}$$

Note the locations of the poles and zeros in the z-plane. Zeros at infinity in the s-plane always map into the z = -1 point. The example illustrate a third-order elliptic-function filter designed using the bilinear transform.

Frequency Transformations

For the design of highpass, bandpass, and band reject filters, a particularly powerful combination consists of using the frequency transformations described in Section elsewhere together with the bilinear transformation. When using this combination, some care must be taken in scaling the specifications properly. This is illustrated by considering the steps in the design of a bandpass filter:

1. First, the lower and upper digital bandedge frequencies are specified as ω_1 and or ω_1, ω_2, ω_3, and ω_4 if an elliptic-function approximation is used.
2. These frequencies are prewarped using [link] to give theband edges of the prototype bandpass analog filter.
3. These frequencies are converted into a single band- edge ω_p or ω_s for the Butterworth and Chebyshev and into ω_p and ω_s for the elliptic-function approximation of the prototype lowpass filter by using Equation 2 from Frequency Transformations and Equation 4 from Frequency Transformations.
4. The lowpass filter is designed for this ω_p and/or ω_s by using one of the four approximations in the sections in Continuous Frequency Definition of Error or some other method.
5. This lowpass analog filter is converted into a bandpass analog filter with the frequency transformation Equation 6 from Frequency Transformations.
6. The bandpass analog filter is then transformed into the desired bandpass digital filter using the bilinear transformation [link].

This is the procedure used in the design Program 8 in the appendix.

When designing a bandpass elliptic-function filter, four frequencies must be specified: the lower stopband edge, the lower passband edge, the upper passband edge, and the upper stopband edge. All four must be prewarped to the equivalent analog values. A problem occurs when the two transition bands of the bandpass filter are converted into the single transition band of the lowpass prototype filter. In general they will be inconsistant; therefore, the narrower of the two transition bands should be used to specify the lowpass filter. The same problem occurs in designing a bandreject elliptic-function filter. Program 8 in the appendix should be studied to understand how this is carried out.

An alternative to the process of converting a lowpass analog into a bandpass analog filter which is then converted into a digital filter, is to first convert the prototype lowpass analog filter into a lowpass digital filter and then make the conversion into a bandpass filter. If the prototype digital filter transfer function is $H_p(z)$ and the frequency transformation is $f(z)$, the desired transformed digital filter is described by
Equation:

$$H(z) = H_p(f(z))$$

Since the frequency response of both $H(z)$ and $H_p(z)$ is obtained by evaluating them on the unit circle in the-z plane, $f(z)$ should map the unit circle onto the unit circle ($|z| = 1 => |f(z)| = 1$). Both $H(z)$ and $H_p(z)$ should be stable; therefore, $f(z)$ should map the interior of the unit circle into the interior of the unit circle ($|z| < 1 => |f(z)| < 1$). If $f(z)$ were viewed as a filter, it would be an “all-pass" filter with a unity magnitude frequency response of the form
Equation:

$$f(z) = \frac{p(z)}{z^n p(1/z)} = \frac{a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0}{a_0 z^n + a_1 z^{n-1} + \cdots + a_n}$$

The prototype digital lowpass filter is usually designed with bandedges at $\pm\pi/2$. Determining the frequency transformation then becomes the problem of solving the $n + 1$ equations
Equation:

$$f\left(e^{j\omega_i}\right) = e^{\pm j\pi/2} = (-1)^i j$$

for the unknown a_k where $i = 0, 1, 2, \cdots n$ and the ω_i are the bandedges of the desired transformed frequency response put in ascending order. The resulting simultaneous equations have a special structure that allow a recursive solution. Details of this approach can be found in [link].

This is an extremely general approach that allows multiple passbands of arbitrary width. If elliptic-function approximations are used, only one of the transition bandwidths can be independently specified. If more than one passband or rejectband is desired, f(z) will be higher order than second order and, therefore, the transformed transfer function $H(f(z))$ will have to be factored using a root finder.

To illustrate the results of using transform methods to design filters, three examples are given which are designed with Program 8 from the appendix.

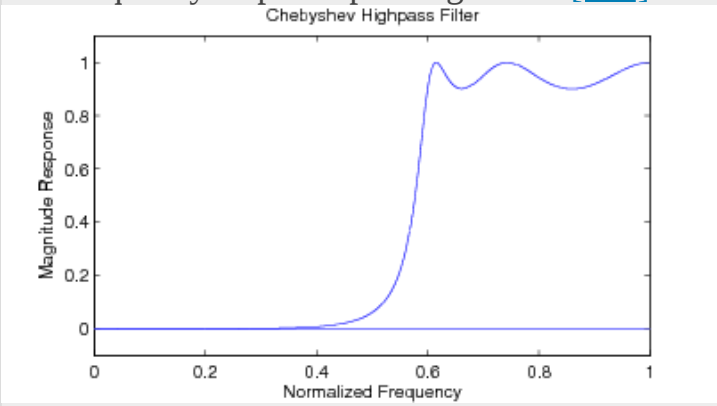
Example:
Design of an Chebyshev Highpass Filter

The specifications are given for a highpass Chebyshev frequency response with a passband edge at $f_p = 0.3$ Hertz with a sampling rate of one sample per second. The order is set at $N = 5$ and the passband ripple at 0.91515 dB. The transfer function is

Equation:

$$H(z) = \frac{(z-1)\left(z^2-2z+1\right)\left(z^2-2z+1\right)}{(z+0.64334)\left(z^2+0.97495z+0.55567\right)\left(z^2+0.57327z+0.83827\right)}$$

The frequency response plot is given in [link].

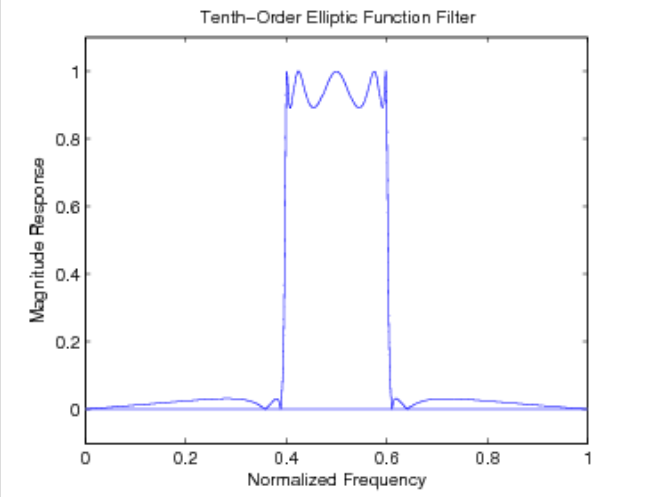


Fifth Order Digital Chebyshev Highpass Filter

Example:

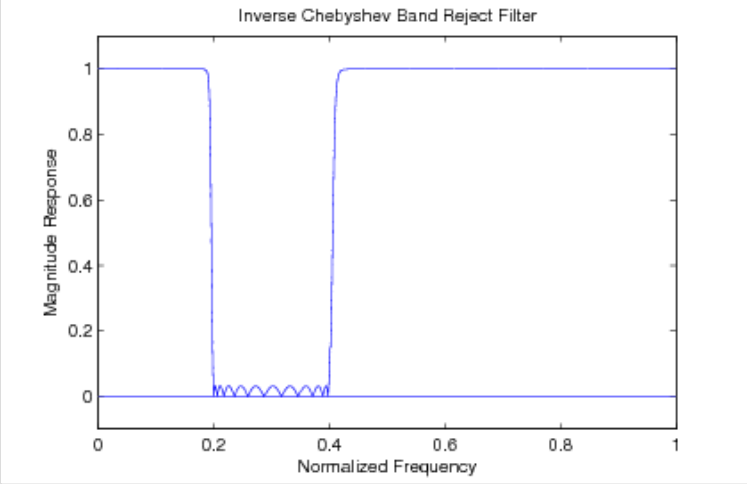
Design of an Elliptic-Function Bandpass Filter

This filter requires a bandpass frequency response with an elliptic-function approximation. The maximum passpand ripple is one dB, the minimum stopband attenuation is 30 dB, the lower stopband edge $f_1 = 0.19$, the lower passband edge $f_2 = 0.2$, the upper passband edge $f_3 = 0.3$, and the upper stopband edge $f_4 = 0.31$ Hertz with a sampling rate of one sample per second. The design program calculated a required prototype order of $N = 6$ and, therefore, a total order of 10. The frequency response plot is shown in [link].



Tenth Order Digital Elliptic Bandpass Filter

Example:
Design of an Inverse-Chebyshev Bandreject Filter
The specifications are given for a bandreject Inverse- Chebyshev frequency response with bandedges at $f_s = 0.1$ and 0.2 Hertz with a sampling rate of one sample per second. The order is set at $N = 11$ and the minimum stopband attenuation at 30 dB. The frequency response plot is given in [link].



Twenty second Order Digital Inverse Chebyshev Band Reject Filter

Summary

This section has described the two most popular and useful methods for transforming a prototype analog filter into a digital filter. The analog frequency variable is used because a literature on analog filter design exists, but more importantly, many approximation theories are more straightforward in terms of the Laplace-transform variable than the z-transform variable. The impulse-invariant method is particularly valuable when time- domain characteristics are important. The bilinear-transform method is the most common when frequency-domain performance is the main interest. Use of the BLT warps the frequency scale and, therefore, the digital band edges must be prewarped to obtain the necessary band edges for the analog filter design. Formulas that transform the analog prototype filters into the desired digital filters and for prewarping specifications were derived.

The use of frequency transformations to convert lowpass filters into highpass, bandpass, and bandreject filters was discussed as a particularly useful combination with the bilinear transformation. These are implemented in Program 8 and design examples from this program were shown.

There are cases where no analytic results are possible or where the desired frequency response is not piecewise constant and transformation methods are not appropriate. Direct methods for these cases are developed in other sections.

Direct Frequency Domain IIR Filter Design Methods

The preceding design methods have been based on designing an analog prototype filter and then converting it to a digital filter. This approach is appropriate for the class of approximations where analytical solutions are possible, but not for many others. In the remaining part of this chapter, methods will be developed that directly design the desired digital filter. Most approaches are extensions of methods used for FIR filters, but they are more complicated for the IIR case where rational approximation is being performed rather than polynomial approximation.

In this section a frequency-sampling design method is developed such that the frequency response of the IIR filter will pass through the given samples of a desired response. Since an IIR filter cannot have linear phase, the sampled response must contain both magnitude and phase. The extension of the frequency- sampling method to a LS-error approximation is not as simple as for the FIR filter. The method presented in this section uses a criterion based on the equation error rather than the more common error between the actual and desired frequency response[link]. Nevertheless, it is a useful noniterative design method. Finally, a general discussion of iterative design methods for LS-frequency response error is given.

Frequency-Sampling Design of IIR Filters

The method for calculating samples of the frequency response of an IIR filter presented in the section on Properties of IIR Filters can be reversed to design a filter much the same way it was for the FIR filter using frequency sampling. The z-transform transfer function for an IIR filter is given by
Equation:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}.$$

The frequency response of the filter is given by setting $z = e^{-j\omega}$. Using the notation
Equation:

$$H(\omega) = H(z)|_{z=e^{-j\omega}}.$$

Equally-spaced samples of the frequency response are chosen so that the number of samples is equal to the number of unknown coefficients in [link]. These $L+1 = M+N+1$ samples of this frequency response are given by
Equation:

$$H_k = H\left(\omega_k\right) = H\left(\frac{2\pi k}{L+1}\right)$$

and can be calculated from the length-$(L+1)$) DFTs of the numerator and denominator.
Equation:

$$H_k = \frac{\mathscr{DFT}\{b_n\}}{\mathscr{DFT}\{a_n\}} = \frac{B_k}{A_k}$$

where the indicated division is term-by-term division for each value of k. Multiplication of both sides of [link] by A_k gives
Equation:

$$B_k = H_k A_k$$

If the length-$(L+1)$ inverse DFT of H_k is denoted by the length- $(L+1)$ sequence h_n, equation [link] becomes cyclic convolution which can be expressed in matrix form by
Equation:

$$\begin{matrix} b_0 \\ b_1 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \end{matrix} = \begin{matrix} h_0 & h_L & h_{L-1} & \cdots & h_1 \\ h_1 & h_0 & h_L & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \\ h_L & & \cdots & & h_0 \end{matrix} \begin{matrix} 1 \\ a_1 \\ \vdots \\ a_N \\ 0 \\ \vdots \\ 0 \end{matrix}$$

Note that the h_n in [link] are not the impulse response values of the filter as used in the FIR case. A more compact matrix notation of is
Equation:

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = [\mathbf{H}] \begin{bmatrix} \mathbf{a} \\ \mathbf{0} \end{bmatrix}$$

where $\mathbf{H}$ is $(L+1)$ by $(L+1)$, $\mathbf{b}$ is length-$(M+1)$, and $\mathbf{a}$ is length-$(N+1)$. Because the lower $L-N$ terms of the right-hand vector of [link] are zero, the $\mathbf{H}$ matrix can be reduced by deleting the right-most $L-N$ columns to give $\mathbf{H}_0$ which causes [link] to become
Equation:

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = [\mathbf{H}_0][\mathbf{a}]$$

Because the first element of $\mathbf{a}$ is unity, it is partitioned to remove the unity term and the remaining length-N vector is denoted $\mathbf{a}^*$. The simultaneous equations represented by [link] are uncoupled by further partitioning of the $\mathbf{H}$ matrix as shown in
Equation:

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_1 & \\ \mathbf{h}_1 & \mathbf{H}_2 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a}^* \end{bmatrix}$$

where $\mathbf{H_1}$ is $(M+1)$ by $(N+1)$, $\mathbf{h}_1$ is length-$(L-M)$, and $\mathbf{H}_2$ is $(L-M)$ by N. The lower $(L-M)$ equations are written
Equation:

$$\mathbf{0} = \mathbf{h}_1 + \mathbf{H}_2 \mathbf{a}^*$$

or
Equation:

$$\mathbf{h}_1 = -\mathbf{H}_2\mathbf{a}^*$$

which must be solved for $\mathbf{a}^*$. The upper $M+1$ equations of [link] are written
Equation:

$$\mathbf{b} = \mathbf{H}_1\mathbf{a}$$

which allows the calculation of $\mathbf{b}$.

If $L = N + M$, $\mathbf{H}_2$ is square. If $\mathbf{H}_2$ is nonsingular, [link] can be solved exactly for the denominator coefficients in $\mathbf{a}^*$, which are augmented by the unity term to give $\mathbf{a}$. From [link], the numerator coefficients in $\mathbf{b}$ are found. If $\mathbf{H}_2$ is singular [link] and there are multiple solutions, a lower order problem can be posed. If there are no solutions, the approximation methods must be used.

Note that any order numerator and denominator can be prescribed. If the filter is in fact an FIR filter, $\mathbf{a}$ is unity and $\mathbf{a}^*$ does not exist. Under these conditions, [link] states that $b_n = h_n$, which is one of the cases of FIR frequency sampling covered [link]. Also note that there is no control over the stability of the filter designed by this method.

Summary

In this section, an interpolation design method was developed and analyzed. Use of the DFT converted the frequency- domain specifications to the time domain. A matrix partitioning allowed uncoupling the solution for the numerator from the solution of the denominator coefficients. The use of the DFT prevents the possibility of unequally spaced frequency samples as was possible for FIR filter design. The solution of simultaneous equations would allow unequal spacing which is not as troublesome as with the FIR filter because IIR filters are usually of lower order.

The frequency-sampling design of IIR filters is somewhat more complicated than for FIR filters because of the requirement that $\mathbf{H}_2$ be nonsingular. As for the FIR filter, the samples of the desired frequency response must satisfy the

conditions to insure that h_n are real. The power of this method is its ability to interpolate arbitrary magnitude and phase specification. In contrast to most direct IIR design methods, this method does not require any iterative optimization with the accompanying convergence problems.

As with the FIR version, because this design approach is an interpolation method rather than an approximation method, the results may be poor between the interpolation points. This usually happens when the desired frequency-response samples are not consistent with what an IIR filter can achieve. One solution to this problem is the same as for the FIR case [link], the use of more frequency samples than the number of filter coefficients and the definition of an approximation error function that can be minimized. There is no simple restriction that will guarantee stable filters. If the frequency-response samples are consistent with an unstable filter, that is what will be designed.

Discrete Least-Squared Equation-Error IIR Filter Design

In order to obtain better practical filter designs, the interpolation scheme of the previous section is extended to give an approximation design method [link]. It should be noted at the outset that the method developed in this section minimizes an equation-error measure and not the usual frequency-response error measure.

The number of frequency samples specified, $L+1$, will be made larger than the number of filter coefficients, $M+N+1$. This means that $\mathbf{H}_2$ is rectangular and, therefore, [link] cannot in general be satisfied. To formulate an approximation problem, a length-$(L+1)$ error vector ε is introduced in [link] and [link] to give
Equation:

$$\begin{bmatrix}\mathbf{b}\\ \mathbf{0}\end{bmatrix} = [\mathbf{H}_0][\mathbf{a}] + [\varepsilon]$$

Equation [link] becomes
Equation:

$$\mathbf{h}_1 - \varepsilon = -\mathbf{H}_2\mathbf{a}^*$$

where now $\mathbf{H}_2$ is rectangular with $(L-M) > N$. Using the same methods as used to derive [link], the error ε is minimized in a least-squared error sense by the solution of the normal equations [link]
Equation:

$$\mathbf{H}_2^T \mathbf{h}_1 = -\mathbf{H}_2^T \mathbf{H}_2 \mathbf{a}^*$$

If the equations are not singular, the solution is
Equation:

$$\mathbf{a}^* = -\left[\mathbf{H}_2^T \mathbf{H}_2\right]^{-1} \mathbf{H}_2^T \mathbf{h}_1.$$

If the normal equations are singular, the pseudo-inverse [link] can be used to obtain a minimum norm or reduced order solution.

The numerator coefficients are found by the same techniques as before in [link]
Equation:

$$\mathbf{b} = \mathbf{H}_1 \mathbf{a}$$

which results in the upper $M+1$ terms in ε being zero and the total squared equation error being minimum.

As is true for LS-error design of FIR filters, [link] is often numerically ill-conditioned and [link] should not be used to solve for $\mathbf{a}^*$. Special algorithms such as those used by Matlab and LINPACK [link], [link] should be employed.

The error ε defined in [link] can better be understood by considering the frequency-domain formulation. Taking the DFT of [link] gives
Equation:

$$B_k = H_k A_k + \varepsilon$$

where ε is the error in trying to satisfy [link] when the equations are over-specified. This can be reformulated in terms of $\mathscr{E}$, the difference between the

frequency response samples of the designed filter and the desired response samples, by dividing [link] by A_k to give

Equation:

$$\mathscr{E}_k = \frac{B_k}{A_k} - H_k = \frac{\varepsilon_k}{A_k}$$

$\mathscr{E}$ is the error in the solution of the approximation problem, and ε is the error in the equations defining the problem. The usual statement of a frequency-domain approximation problem is in terms of minimizing some measure of $\mathscr{E}$, but that results in solving nonlinear equations. The design procedure developed in this section minimizes the squared error ε, thus only requiring the solution of linear equations. There is an important relation between these problems. [link] shows that minimizing ε is the same as minimizing $\mathscr{E}$ weighted by A. However, A is unknown until after the problem is solved.

Although this is posed as a frequency-domain design method, the method of solution for both the interpolation problem and the LS equation-error problem is the same as the time-domain Prony's method, discussed in Complex and Minimum Phase Approximation of reference [link].

Numerous modifications and extensions can be made to this method. If the desired frequency response is close to what can be achieved by an IIR filter, this method will give a design approximately the same as that of a true least-squared solution-error method. It can be shown that $\varepsilon = 0 \leftrightarrow \mathscr{E} = 0$. In some cases, improved results can be obtained by estimating A_k and using that as a weight on ε to approximate minimizing $\mathscr{E}$. There are iterative methods based on solving [link] and [link] to obtain values for A_k. These values are used as weights on ε to solve for a new set of A_k used as a new set of weights to solve again for A_k[link][link]. We found this approach to converge slowly, but a recent paper using the log-magnitude [link] was more successful. Other approaches are given in [link], [link], [link]. The solution of [link] and [link] is sometimes used to obtain starting values for other iterative optimization algorithms that need good starting values for convergence.

An interesting iterative design algorithm that can design to approximate complex or magnitude frequency responses has be recently proposed by Jackson [link]. A different approach to the same problem was posed by Soewito [link], [link].

To illustrate this design method a sixth-order lowpass filter was designed with 41 frequency samples to approximate. The magnitude of those less than 0.2 Hz is one and of those greater than 0.2 is zero. The phase was experimentally adjusted to result in a good magnitude response. The design was performed with Program 9 in the appendix of [link] and the frequency response is shown in Figure 7-33 of [link]. Matlab programs have recently been written which are smaller and easier to understand than those in FORTRAN.

Summary

In this section an LS-error approximation method was posed to design IIR filters. By using an equation-error rather than a solution-error criterion, a problem resulted that required only the solution of simultaneous linear equations.

Like the FIR filter version, the IIR frequency sampling design method and the LS equation-error extension can be used for complex approximation and, therefore, can design with both magnitude and phase specifications.

If the desired frequency-response samples are close to what an IIR filter of the specified order can achieve, this method will produce a filter very close to what a true least-squared error method would. However, when the specifications are not consistent with what can be achieved and the approximating error is large, the results can be very poor and in some cases, unstable. It is particularly difficult to set realistic phase response specifications. With this method, it is even more important to have a design environment that will allow easy trial-and-error procedure.

Newly published works which will be discussed here are [link], [link], [link], [link], [link], [link], [link]. Other references can be found in [link], [link], [link], [link], [link], [link]. The Matlab command `invfreqz()` which is an inverse to the `freqz()` command gives a similar or, perhaps, the same result as the method described in this note but uses a different formulation [link], [link].

more

Practical problems occur in the design of a filter to separate signals according to their energy. Because the energy content of a signal is the integral or sum of the square of the signal, a mean-squared-error measure is natural. Unfortunately, for the IIR filter design problem, the optimization procedure is nonlinear. This was pointed out in the last section where the equation error was used in order to have a linear problem.

Because of the nonlinear nature of the least-squared-error minimization, the method of solution becomes dependent on the desired frequency response, and therefore, there is no single method for design. The mean-squared error for magnitude approximation is defined as
Equation:

$$q\left(x\right)=\sum_{i=0}^{L}|H\left(\omega_{i}\right)|-|H_{d}\left(\omega_{i}\right)^{2}$$

where x is a vector of filter parameters chosen to minimize q, and the error is sampled at $L+1$ frequencies ω_{i}. Steiglitz [link] chose the parameter vector x to be the coefficients of a cascade structure in order to best fit an iterative optimization scheme. He applied a standard optimization algorithm, the Fletcher-Powell method, to the minimization of [link]. Other methods which are more directly related to a squared-error measure can also be used.

Practical difficulties exist in solving this approximation problem. In some cases, local minima are found rather than the global minimum. In other cases, convergence of the minimization algorithm is slow or does not occur at all. Numerical problems can result from ill-conditioned equations, and there is no guarantee that the designed filter will be stable.

An important factor is the choice of a desired frequency- response function $H_{d}\left(\omega\right)$ that does not result in the optimum approximation having a large error. This often means not having an abrupt discontinuity between the passband and stopband.

Another factor is the starting of the iterative optimization algorithm with a set of coefficients in x that is close to the optimum. This can be accomplished by using the frequency sampling method to give a design that can be used to start a least-squares algorithm. Because the error defined in [link] is in terms of

magnitudes, an unstable design can be converted to a stable one by moving the unstable pole at a radius of r in the z-plane to a radius of $1/r$. This does not change the magnitude frequency response and does stabilize the effect of that pole [link].

A generalization of the idea of a squared-error measure is defined by raising the error to the p power where p is a positive integer. This error is defined by
Equation:

$$q\left(x\right) = \sum_{i=0}^{L} \left|H\left(\omega\right) - H_{d}\left(\omega\right)\right|^{p}$$

Deczky [link] developed this approach and used the Fletcher-Powell method to minimize [link]. He also applied this method to the approximation of a desired group-delay function. An important characteristic of this formulation is that the solution approaches the Chebyshev or mini-max solution as p becomes large. Initial work shows the method of iteratively reweighted least squared error (IRLS) as was applied to the FIR filter design in [link] can also be used for L_p and constrained least squared error optimal design of IIR filters [link].

The Chebyshev Error Criterion for IIR Filter Design

The error measure that often best meets filter design specifications is the maximum error in the frequency response that occurs over a band. The filter design problem becomes the problem of minimizing the maximum error (the min-max problem).

Among several approaches to this error minimization, one is by Deczky which minimizes the p-power error of [link] for large p. Generally, $p = 10$ or greater approximates a Chebyshev result [link]. Another is by Dolan and Kaiser which uses a penalty-function approach.

Linear programming can be applied to this error measure by linearizing the equations in much the same way as in [link] [link]. In contrast to the FIR case, this can be a practical design method because the order of practical IIR filters is generally much lower than for FIR filters. A scheme called differential correction has also proven to be effective.

Although the rational approximation problem is nonlinear, an application of the Remes exchange algorithm can be implemented [link]. Since the zeros of the numerator of the transfer function mainly control the stopband characteristics of a filter, and the zeros of the denominator mainly control the passband, the effects of the two are somewhat uncoupled. An application of the Remes exchange algorithm, alternating between the numerator and denominator, gives an effective method for designing IIR filters with a Chebyshev error criterion. If the order of the numerator and denominator are the same and the desired filter is an ideal lowpass filter, the Remes exchange should give the same result as the elliptic function filter. However, this approach allows any order numerator or denominator to be set and any shape passband to be approximated. There are cases where a lower-order denominator than numerator results in a filter with fewer required muliplications than an elliptic-function filter [link].

Prony's Method for Time-Domain Design of IIR Filters

The problem of designing an IIR digital filter with a prescribed time-domain response is addressed in this section. Most formulations of time-domain design of IIR filters result in nonlinear equations for the same reasons as for frequency-domain design. Prony, in 1790, derived a special formulation for the analysis of elastic properties of gases, which resulted in linear equations. A more general form of Prony's method can be applied to the IIR filter design by use of a matrix description [link], [link].

The transfer function of an IIR filter is given by
Equation:

$$H\left(z\right) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} = h_0 + h_1 z^{-1} + h_2 z^{-2} + \cdots$$

and the impulse response $h(n)$ is related to $H(z)$ by the z transform.
Equation:

$$H\left(z\right) = \sum_{n=0}^{\infty} \; h\left(n\right) \; z^{-n}$$

[link] can be written
Equation:

$$B(z) = H(z)\ A(z)$$

which is the z-transform version of convolution. This convolution can be written as a matrix multiplication. Using the first K+1 terms of the impulse response, this is written
Equation:

$$\begin{matrix} b_0 \\ b_1 \\ \vdots \\ b_M \\ 0 \\ \vdots \\ 0 \end{matrix} = \begin{matrix} & & & & & \\ h_0 & 0 & 0 & \cdots & 0 \\ h_1 & h_0 & 0 & & \\ h_2 & h_1 & h_0 & & \\ \vdots & & & & \vdots \\ h_L & & \cdots & & h_0 \\ & & & & \end{matrix} \begin{matrix} 1 \\ a_1 \\ \vdots \\ a_N \\ 0 \\ \vdots \\ 0 \end{matrix}$$

In order to uncouple the calculations of the a_n and the b_n, the matrices are partitioned to give
Equation:

$$\begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{H_1} \\ \mathbf{h}_1 \quad \mathbf{H}_2 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{a}^* \end{bmatrix}$$

where $\mathbf{b}$ is the vector of the $M+1$ numerator coefficients of [link], $\mathbf{a}^*$ is the vector of the N denominator coefficients ($ao = 1$), $\mathbf{h}_1$ is the vector of the last $(K-M)$ terms of the impulse response, $\mathbf{H_1}$ is the $M+1$ by $N+1$ partition of [link], and $\mathbf{H}_2$ is the $(K-M)$ by N remaining part. The lower $K-M$ equations are written
Equation:

$$\mathbf{0} = \mathbf{h}_1 + \mathbf{H}_2\mathbf{a}^*$$

or
Equation:

$$\mathbf{h}_1 = -\mathbf{H}_2\mathbf{a}^*$$

which must be solved for $\mathbf{a}^*$. The upper $M+1$ equations of [link] are written
Equation:

$$\mathbf{b} = \mathbf{H}_1\mathbf{a}$$

which allows the calculation of $\mathbf{b}$.

If $L = N + M$, then $\mathbf{H}_2$ is square. If $\mathbf{H}_2$ is nonsingular, [link] can be solved exactly for the denominator coefficients in $\mathbf{a}^*$, which are augmented by the unity term to give $\mathbf{a}$. From [link], the numerator coefficients in $\mathbf{b}$ are found. If $\mathbf{H}_2$ is singular [link] and there are multiple solutions, a lower order problem can be posed. If there are no solutions, the methods of the next section must be used.

Note that any order numerator and denominator can be prescribed. If the filter is in fact an FIR filter, $\mathbf{a}$ is unity and $\mathbf{a}^*$ does not exist. Under these conditions, [link] states that $b_n = h_n$, which is one of the cases of FIR frequency sampling covered in Section 3.1 of [link]. Also note that there is no control over the stability of the filter designed by this method.

Although Prony's method, applied to the time-domain design problem here, is similar to the solution of the frequency-sampling IIR design problem, there are important differences. The inverse DFT is used to obtain the matrix in the frequency domain problem, which is cyclic convolution. Equation [link] is noncyclic convolution and the $K+1$ terms of $h(n)$, used to form $\mathbf{H}$, result from a truncation of the infinitely long sequence.

An Approximate Solution or the Least Equation Error Problem

In order to obtain better practical filter designs, the interpolation scheme of the previous section is extended to give an approximation design method [link]. It

should be noted at the outset that the method developed in this section minimizes an equation-error measure and not the usual frequency-response error measure.

The number of samples specified, $L+1$, will be made larger than the number of filter coefficients, $M+N+1$. This means that $\mathbf{H}_2$ is rectangular and, therefore, [link] cannot in general be satisfied. To formulate an approximation problem, a length-$(L+1)$ error vector ε is introduced in [link] and [link] to give

Equation:

$$\begin{bmatrix}\mathbf{b}\\ \mathbf{0}\end{bmatrix} = [\mathbf{H}_0][\mathbf{a}] + [\varepsilon]$$

[link] becomes

Equation:

$$\mathbf{h}_1 - \varepsilon = -\mathbf{H}_2\mathbf{a}^*$$

where now $\mathbf{H}_2$ is rectangular with $(L-M) > N$. Using the same methods as used to derive [link], the error ε is minimized in a least-squared error sense by the solution of the normal equations [link]

Equation:

$$\mathbf{H}_2^T\mathbf{h}_1 = -\mathbf{H}_2^T\mathbf{H}_2\mathbf{a}^*$$

If the equations are not singular, the solution is

Equation:

$$\mathbf{a}^* = -\left[\mathbf{H}_2^T\mathbf{H}_2\right]^{-1}\mathbf{H}_2^T\mathbf{h}_1.$$

If the normal equations are singular, the pseudo-inverse [link] can be used to obtain a minimum norm or reduced order solution.

The numerator coefficients are found by the same techniques as before in [link]

Equation:

$$\mathbf{b} = \mathbf{H}_1\mathbf{a}$$

which results in the upper $M + 1$ terms in ε being zero and the total squared equation error being minimum.

As is true for LS-error design of FIR filters, [link] is often numerically ill-conditioned and [link] should not be used to solve for $\mathbf{a}^*$. Special algorithms such as those used by Matlab and LINPACK [link], [link] should be employed.

Various modifications can be made to the form of Prony's method presented. After the denominator is found by minimizing the equation error, the numerator can be found by minimizing the solution error. It is possible to mix the exact and approximate methods. The details can be found in [link], [link], [link].

Several modifications to Prony's method have been made to use it to minimize the solution error. Most of these iteratively minimize a weighted-equation error with Prony's method and update the weights from the previous determination of a[link], [link].

If an LS-error, time-domain approximation is the desired result, a minimization technique can be applied directly to the solution error. The most successful method seems to be the Gauss- Newton algorithm with a step-size control. This combined with Prony's method to find starting parameters is an effective design tool.

Block, Multi-rate, Multi-dimensional Processing and Distributed Arithmetic

Introduction

The partitioning of long or infinite strings of data into shorter sections or blocks has been used to allow application of the FFT to realize on-going or continuous convolution [link], [link]. These notes develop the idea of block processing and shows that it is a generalization of the overlap-add and overlap-save methods [link], [link]. They further generalize the idea to a multidimensional formulation of convolution [link], [link]. Moving in the opposite direction, it is shown that, rather than partitioning a string of scalars into blocks and then into blocks of blocks, one can partition a scalar number into blocks of bits and then include the operation of multiplication in the signal processing formulation. This is called distributed arithmetic [link] and, since it describes operations at the bit level, is completely general. These notes try to present a coherent development of these ideas.

Block Signal Processing

In this section the usual convolution and recursion that implements FIR and IIR discrete-time filters are reformulated in terms of vectors and matrices. Because the same data is partitioned and grouped in a variety of ways, it is important to have a consistent notation in order to be clear. The n^{th} element of a data sequence is expressed $h(n)$ or, in some cases to simplify, h_n. A block or finite length column vector is denoted h_n with n indicating the n^{th} block or section of a longer vector. A matrix, square or rectangular, is indicated by an upper case letter such as H with a subscript if appropriate.

Block Convolution

The operation of a finite impulse response (FIR) filter is described by a finite convolution as
Equation:

$$y(n) = \sum_{k=0}^{L-1} h(k)\, x(n-k)$$

where $x(n)$ is causal, $h(n)$ is causal and of length L, and the time index n goes from zero to infinity or some large value. With a change of index variables this becomes
Equation:

$$y(n) = \sum_{k=0}^{n} h(n-k)\, x(k)$$

which can be expressed as a matrix operation by
Equation:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 & x_0 \\ h_1 & h_0 & 0 & & & x_1 \\ h_2 & h_1 & h_0 & & & x_2 \\ \vdots & \vdots & & & \vdots & \vdots \end{bmatrix}.$$

The H matrix of impulse response values is partitioned into N by N square sub matrices and the X and Y vectors are partitioned into length-N blocks or sections. This is illustrated for $N = 3$ by
Equation:

$$H_0 = \begin{bmatrix} h_0 & 0 & 0 \\ h_1 & h_0 & 0 \\ h_2 & h_1 & h_0 \end{bmatrix} \qquad H_1 = \begin{bmatrix} h_3 & h_2 & h_1 \\ h_4 & h_3 & h_2 \\ h_5 & h_4 & h_3 \end{bmatrix} \qquad \text{etc.}$$

Equation:

$$x_0 = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \qquad x_1 = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \qquad \underline{y}_0 = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} \qquad \text{etc.}$$

Substituting these definitions into [link] gives
Equation:

$$\begin{bmatrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} H_0 & 0 & 0 & \cdots & 0 & x_0 \\ H_1 & H_0 & 0 & & & x_1 \\ H_2 & H_1 & H_0 & & & x_2 \\ \vdots & \vdots & & & \vdots & \vdots \end{bmatrix}$$

The general expression for the n^{th} output block is
Equation:

$$\underline{y}_n = \sum_{k=0}^{n} H_{n-k}\, x_k$$

which is a vector or block convolution. Since the matrix-vector multiplication within the block convolution is itself a convolution, [link] is a sort of convolution of convolutions and the finite length matrix-vector multiplication can be carried out using the FFT or other fast convolution methods.

The equation for one output block can be written as the product
Equation:

$$\underline{y}_2 = \begin{bmatrix} H_2 H_1 H_0 \end{bmatrix} \begin{matrix} x_0 \\ x_1 \\ x_2 \end{matrix}$$

and the effects of one input block can be written
Equation:

$$\begin{matrix} H_0 \\ H_1 \\ H_2 \end{matrix} \; x_1 = \begin{matrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \end{matrix} \; .$$

These are generalize statements of overlap save and overlap add [link], [link]. The block length can be longer, shorter, or equal to the filter length.

Block Recursion

Although less well-known, IIR filters can be implemented with block processing [link], [link], [link], [link], [link]. The block form of an IIR filter is developed in much the same way as for the block convolution implementation of the FIR filter. The general constant coefficient difference equation which describes an IIR filter with recursive coefficients a_l, convolution coefficients b_k, input signal $x(n)$, and output signal $y(n)$ is given by
Equation:

$$y\,(n) = \sum_{l=1}^{N-1} a_l\, y_{n-l} + \sum_{k=0}^{M-1} b_k\, x_{n-k}$$

using both functional notation and subscripts, depending on which is easier and clearer. The impulse response $h(n)$ is

Equation:

$$h\left(n\right) = \sum_{l=1}^{N-1} a_l \, h\left(n-l\right) + \sum_{k=0}^{M-1} b_k \, \delta\left(n-k\right)$$

which can be written in matrix operator form

Equation:

$$\begin{matrix} 1 & 0 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & & \\ a_2 & a_1 & 1 & & \\ a_3 & a_2 & a_1 & & \\ 0 & a_3 & a_2 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ \vdots \end{matrix} = \begin{matrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ 0 \\ \vdots \end{matrix}$$

In terms of N by N submatrices and length-N blocks, this becomes

Equation:

$$\begin{matrix} A_0 & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & 0 & & \\ 0 & A_1 & A_0 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} h_0 \\ h_1 \\ h_2 \\ \vdots \end{matrix} = \begin{matrix} \underline{b}_0 \\ \underline{b}_1 \\ 0 \\ \vdots \end{matrix}$$

From this formulation, a block recursive equation can be written that will generate the impulse response block by block.

Equation:

$$A_0 \, h_n + A_1 \, h_{n-1} = 0 \text{ for } n \geq 2$$

Equation:

$$h_n = -A_0^{-1} A_1 \, h_{n-1} = K \, h_{n-1} \text{ for } n \geq 2$$

with initial conditions given by
Equation:

$$h_1 = -A_0^{-1} A_1 A_0^{-1} \underline{b}_0 + A_0^{-1} \underline{b}_1$$

This can also be written to generate the square partitions of the impulse response matrix by
Equation:

$$H_n = K H_{n-1} \text{ for } n \geq 2$$

with initial conditions given by
Equation:

$$H_1 = K A_0^{-1} B_0 + A_0^{-1} B_1$$

ane $K = -A_0^{-1} A_1$. This recursively generates square submatrices of H similar to those defined in [link] and [link] and shows the basic structure of the dynamic system.

Next, we develop the recursive formulation for a general input as described by the scalar difference equation [link] and in matrix operator form by
Equation:

$$\begin{matrix} 1 & 0 & 0 & \cdots & 0 \\ a_1 & 1 & 0 & & \\ a_2 & a_1 & 1 & & \\ a_3 & a_2 & a_1 & & \\ 0 & a_3 & a_2 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \end{matrix} = \begin{matrix} b_0 & 0 & 0 & \cdots & 0 \\ b_1 & b_0 & 0 & & \\ b_2 & b_1 & b_0 & & \\ 0 & b_2 & b_1 & & \\ 0 & 0 & b_2 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \end{matrix}$$

which, after substituting the definitions of the sub matrices and assuming the block length is larger than the order of the numerator or denominator, becomes
Equation:

$$\begin{matrix} A_0 & 0 & 0 & \cdots & 0 \\ A_1 & A_0 & 0 & & \\ 0 & A_1 & A_0 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} \underline{y}_0 \\ \underline{y}_1 \\ \underline{y}_2 \\ \vdots \end{matrix} = \begin{matrix} B_0 & 0 & 0 & \cdots & 0 \\ B_1 & B_0 & 0 & & \\ 0 & B_1 & B_0 & & \\ \vdots & & & \vdots & \end{matrix} \begin{matrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{matrix} .$$

From the partitioned rows of [link], one can write the block recursive relation
Equation:

$$A_0 \, \underline{y}_{n+1} + A_1 \, \underline{y}_n = B_0 \, x_{n+1} + B_1 \, x_n$$

Solving for $\underline{y}_{n+1}$ gives
Equation:

$$\underline{y}_{n+1} = -A_0^{-1} A_1 \, \underline{y}_n + A_0^{-1} B_0 \, x_{n+1} + A_0^{-1} B_1 \, x_n$$

Equation:

$$\underline{y}_{n+1} = K \, \underline{y}_n + H_0 \, x_{n+1} + \tilde{H}_1 \, x_n$$

which is a first order vector difference equation [link], [link]. This is the fundamental block recursive algorithm that implements the original scalar difference equation in [link]. It has several important characteristics.

1. The block recursive formulation is similar to a state variable equation but the states are blocks or sections of the output [link], [link], [link], [link].
2. The eigenvalues of K are the poles of the original scalar problem raised to the N power plus others that are zero. The longer the block length, the “more stable" the filter is, i.e. the further the poles are from the unit circle [link], [link], [link], [link], [link].
3. If the block length were shorter than the denominator, the vector difference equation would be higher than first order. There would be a non zero A_2. If the block length were shorter than the numerator, there would be a non zero B_2 and a higher order block convolution operation. If the block length were one, the order of the vector equation would be the same as the scalar equation. They would be the same equation.
4. The actual arithmetic that goes into the calculation of the output is partly recursive and partly convolution. The longer the block, the more the output is calculated by convolution and, the more arithmetic is required.

5. It is possible to remove the zero eigenvalues in K by making K rectangular or square and N by N This results in a form even more similar to a state variable formulation [link], [link]. This is briefly discussed below in The Z-Transform.
6. There are several ways of using the FFT in the calculation of the various matrix products in [link] and in [link] and [link]. Each has some arithmetic advantage for various forms and orders of the original equation. It is also possible to implement some of the operations using rectangular transforms, number theoretic transforms, distributed arithmetic, or other efficient convolution algorithms [link], [link], [link], [link], [link], [link].
7. By choosing the block length equal to the period, a periodically time varying filter can be made block time invariant. In other words, all the time varying characteristics are moved to the finite matrix multiplies which leave the time invariant properties at the block level. This allows use of z-transform and other time-invariant methods to be used for stability analysis and frequency response analysis [link], [link]. It also turns out to be related to filter banks and multi-rate filters [link], [link], [link].

Block State Formulation

It is possible to reduce the size of the matrix operators in the block recursive description [link] to give a form even more like a state variable equation [link], [link], [link]. If K in [link] has several zero eigenvalues, it should be possible to reduce the size of K until it has full rank. That was done in [link] and the result is

Equation:

$$\underline{z}_n = K_1 \, \underline{z}_{n-1} + K_2 \, x_n$$

Equation:

$$\underline{y}_n = H_1 \, \underline{z}_{n-1} + H_0 \, x_n$$

where H_0 is the same N by N convolution matrix, N_1 is a rectangular L by N partition of the convolution matrix H, K_1 is a square N by N matrix of full rank, and K_2 is a rectangular N by L matrix.

This is now a minimal state equation whose input and output are blocks of the original input and output. Some of the matrix multiplications can be carried out using the FFT or other techniques.

Block Implementations of Digital Filters

The advantage of the block convolution and recursion implementations is a possible improvement in arithmetic efficiency by using the FFT or other fast convolution methods for some of the multiplications in [link] or [link] [link], [link]. There is the reduction of quantization effects due to an effective decrease in the magnitude of the eigenvalues and the possibility of easier parallel implementation for IIR filters. The disadvantages are a delay of at least one block length and an increased memory requirement.

These methods could also be used in the various filtering methods for evaluating the DFT. This the chirp z-transform, Rader's method, and Goertzel's algorithm.

Multidimensional Formulation

This process of partitioning the data vectors and the operator matrices can be continued by partitioning [link] and [link] and creating blocks of blocks to give a higher dimensional structure. One should use index mapping ideas rather than partitioned matrices for this approach [link], [link].

Periodically Time-Varying Discrete-Time Systems

Most time-varying systems are periodically time-varying and this allows special results to be obtained. If the block length is set equal to the period of the time variations, the resulting block equations are time invariant and all to the time varying characteristics are contained in the matrix multiplications. This allows some of the tools of time invariant systems to be used on periodically time-varying systems.

The PTV system is analyzed in [link], [link], [link], [link], the filter analysis and design problem, which includes the decimation–interpolation structure, is addressed in [link], [link], [link], and the bandwidth compression problem in [link]. These structures can take the form of filter banks [link].

Multirate Filters, Filter Banks, and Wavelets

Another area that is related to periodically time varying systems and to block processing is filter banks [link], [link]. Recently the area of perfect reconstruction filter banks has been further developed and shown to be closely related to wavelet based signal analysis [link], [link], [link], [link], [link]. The filter bank structure has several forms with the polyphase and lattice being particularly interesting. Further work on multirate filters can be found in [link], [link], [link], [link], [link], [link].

An idea that has some elements of multirate filters, perfect reconstruction, and distributed arithmetic is given in [link], [link]. Parks has noted that design of multirate filters has some elements in common with complex approximation and of 2-D filter design [link], [link] and is looking at using Tang's method for these designs.

Distributed Arithmetic

Rather than grouping the individual scalar data values in a discrete-time signal into blocks, the scalar values can be partitioned into groups of bits. Because multiplication of integers, multiplication of polynomials, and discrete-time convolution are the same operations, the bit-level description of multiplication can be mixed with the convolution of the signal processing. The resulting structure is called distributed arithmetic [link], [link]. It can be used to create an efficient table look-up scheme to implement an FIR or IIR filter using no multiplications by fetching previously calculated partial products which are stored in a table. Distributed arithmetic, block processing, and multi-dimensional formulations can be combined into an integrated powerful description to implement digital filters and processors. There may be a new form of distributed arithmetic using the ideas in [link], [link].